Instytut Badań Systemowych
Polskiej Akademii Nauk

Streszczenie rozprawy doktorskiej

Wielokryterialne podejście do rozwiązywania zagadnienia załadunku z wyborem

mgr inż. Przemysław Pyzel

Promotor
prof. dr hab. Ewa Bednarczuk
Promotor pomocniczy
dr Janusz Miroforidis

Warszawa, 2020

Cel i główne wyniki rozprawy

Cel. Celem rozprawy jest wykorzystanie technik optymalizacji wielokryterialnej do rozwiązywania zagadnienia załadunku z wyborem (ang. *Multiple-Choice Knapsack Problem*) (problemu ($MCKP$)) oraz konstrukcji dokładnego (nie heurystycznego), efektywnego algorytmu `BISSA`, który znajduje dokładne rozwiązanie problemu ($MCKP$), albo:

a) rozwiązanie przybliżone problemu ($MCKP$),

b) wartość profitu rozwiązania przybliżonego (wartość funkcji celu w tym problemie),

c) oszacowanie od góry na wartość optymalnego profitu problemu ($MCKP$).

Ponadto celem rozprawy jest konstrukcja algorytmu dokładnego i algorytmu heurystycznego, które:

a) algorytm dokładny – startując z uzyskanego rozwiązania przybliżonego znajduje rozwiązanie dokładne problemu ($MCKP$),

b) algorytm heurystyczny – stosując techniki heurystyczne może poprawić uzyskane rozwiązanie przybliżone problemu ($MCKP$).

Teza. Zastosowanie technik optymalizacji wielokryterialnej do rozwiązywania zagadnienia ($MCKP$), pozwala skonstruować relatywnie prosty algorytm, który daje wyniki porównywalne z wynikami otrzymanymi przez komercyjne pakiety do rozwiązywania zadań programowania całkowitoliczbowego (np. GUROBI)

Struktura rozprawy i główne wyniki. Rozprawa podzielona jest na dwie części. Część I dotyczy związków zagadnienia załadunku z wyborem i odpowiednich zagadnień optymalizacji dwu-kryteriowej. W części tej udowodnione są nowe fakty, niezbędne do sformułowania metody `BISSA` (ang. *Bi-objective Approximate Solution Search Algorithm*). Część ta kończy się sformułowaniem metody `BISSA` oraz omówieniem jej podstawowych własności. Część I składa się z czterech rozdziałów. W rozdziale 1 zdefiniowane jest zagadnienie załadunku z wyborem ($MCKP$) i przedstawiony jest aktualny stan badań w zakresie algorytmów rozwiązujących tę klasę problemów. W rozdziale 2 sformułowany jest liniowy dwu-kryteriowy problem optymalizacyjny (BP) powiązany z oryginalnym problemem ($MCKP$), oraz zbadane są relacje pomiędzy tymi problemami. Rozdział 3 zawiera nowe wyniki, niezbędne do wprowadzenia metody `BISSA`. Przeanalizowane są związki pomiędzy problemem ($MCKP$) oraz problemem (BP) i problemem zeskalaryzowanym ($BS(\lambda)$), będącym liniową skalaryzacją problemu (BP), zależnym od współczynników λ. W szczególności, w podrozdziale 3.2 wyprowadzone zostały jawne formuły na rozwiązanie problemu ($BS(\lambda)$). Rozdział 4, stanowiący główny wynik rozprawy, zawiera sformułowanie i analizę nowej metody (algorytmu) `BISSA` do znajdowania, w ogólnym przypadku, rozwiązania przybliżonego problemu ($MCKP$), w oparciu o podejście wielokryterialne, z wykorzystaniem wyników teoretycznych uzyskanych w poprzednich rozdziałach. Algorytm `BISSA` rozwiązuje, w ogólnym przypadku, w sposób przybliżony problem ($MCKP$) poprzez rozwiązanie serii zeskalaryzowanych liniowych dwu-kryteriowych problemów ($BS(\lambda)$), przy czym rozwiązanie problemu ($BS(\lambda)$ dane jest w postaci jawnej formuły.

Główną zaletą proponowanej metody `BISSA` jest to, że zeskalaryzowane liniowe dwukryteriowe zadanie ($BS(\lambda)$) może być precyzyjnie, efektywnie i szybko rozwiązane poprzez

wykorzystanie struktury zbioru dopuszczalnego X tego problemu oraz podanych formuł. Mianowicie, te zeskalaryzowane problemy mogą być zdekomponowane do k niezależnych podproblemów, których rozwiązania są otrzymywane przez proste, jawne formuły. Ta właściwość proponowanej metody jest szczególnie przydatna w obliczeniach równoległych, pozwala generować rozwiązania zeskalaryzowanych problemów szybko i efektywnie.

Otrzymane w wyniku działania algorytmu `BISSA` (o ile nie wyznaczył on rozwiązania dokładnego) przybliżone rozwiązanie $\hat{x}$, może posłużyć jako punkt startowy dla innych, np. heurystycznych lub dokładnych, algorytmów wyznaczających rozwiązanie problemu $(MCKP)$. W rozdziale 5 zaproponowano dwa kolejne algorytmy, poprawiające wyniki dostarczane przez algorytm `BISSA`: dokładny algorytm `SKAN-DOK` i heurystyczny algorytm `SKAN-LIDER`.

Część II dotyczy aspektów obliczeniowych algorytmu `BISSA` i algorytmu `SKAN-LIDER`. Eksperymenty pokazują, że w zależności od rodzaju danych, proponowana metoda generuje relatywnie szybko element $\hat{x}$, który jest przybliżonym rozwiązaniem problemu $(MCKP)$. Tym samym otrzymywane jest oszacowanie od dołu na wartość optymalnego profitu (LB) (ang. *lower bound*) i oszacowanie od góry na wartość optymalnego profitu (UB) (ang. *upper bound*).

Algorytm `BISSA` został poddany testom. Przeprowadzono testy tego algorytmu na problemach nieskorelowanych (*unc* – *uncorrelated*) np. (*unc*, 10, 1000), (*unc*, 100, 100), (*unc*, 1000, 100), słabo skorelowanych (*wco* – *weakly correlated*) np. (*wco*, 10, 10), a także na problemach wielkiej skali (*unc*, 1000, 1000), gdzie w oznaczeniu typu problemu, pierwsza liczba oznacza liczbę kategorii w problemie $(MCKP)$, a druga – liczbę przedmiotów w każdej z kategorii. Rozdział 6 zawiera wyniki testów oraz ich porównanie z wynikami otrzymanymi za pomocą innych znanych algorytmów: `EXACT` (algorytm znajdujący wartość profitu rozwiązania optymalnego), `GREEDY` (algorytm wyznaczający optymalne rozwiązanie ciągłej relaksacji problemu $(MCKP)$, dający wartość oszacowania od góry na wartość optymalnego profitu). Ponadto wyniki działania algorytmu `BISSA` zostały porównane z wynikami otrzymanymi za pomocą pakietu GUROBI.

Część I
Zagadnienie załadunku z wyborem

1 Zagadnienie załadunku z wyborem

Zagadnienie plecakowe (ang. *Knapsack Problem*) (KP) jest szczególną postacią zagadnienia programowania całkowitoliczbowego. Pierwszą analizą problemu (KP) była praca [24]. Zagadnieniu (KP) poświęcone są monografie [17, 23]. Zagadnienie załadunku z wyborem (ang. *Multiple–Choice Knapsack Problem*) – problem $(MCKP)$ – jest pewnym szczególnym przypadkiem zagadnienia (KP). Problem $(MCKP)$ po raz pierwszy opisano w późnych latach 70– tych XX wieku [33].

Problem $(MCKP)$ jest sformułowany następująco. Danych jest k zbiorów N_1, N_2,...,N_k, o liczności $|N_i| = n_i$, $i = 1, ..., k$. Każdemu elementowi każdego zbioru przypisano nieujemny "profit" (ang. *profit*) $p_{ij} \geqslant 0$ i "koszt" (ang. *cost*) $c_{ij} \geqslant 0$, $i = 1, ..., k$, $j = 1, ..., n_i$. Liczby p_{ij} oraz c_{ij}, $i = 1, ..., k$, $j = 1, ..., n_i$ nie muszą być całkowite.

Niech x_{ij}, $i = 1, ..., k$, $j = 1, ..., n_i$, będą zmiennymi decyzyjnymi zdefiniowanymi

następująco

$$x_{ij} = \begin{cases} 1 & \text{jeżeli element } j \text{ zbioru } N_i \text{ jest wybrany} \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Wszystkie x_{ij} tworzą wektor x o długości $n = \sum_{i=1}^{k} n_i$, $x \in \mathbb{R}^n$, co zapisujemy

$$x := (x_{11}, x_{12}, ..., x_{1n_1}, x_{21}, ..., x_{2n_2}, ..., x_{k1}, x_{k2}, ...x_{kn_k})^T = (x_{ij})_{\substack{1 \leqslant i \leqslant k \\ 1 \leqslant j \leqslant n_i}}.$$

Przy powyższych oznaczeniach, problem $(MCKP)$ jest postaci

$$(MCKP) \quad \begin{array}{l} \max \ \sum_{i=1}^{k} \sum_{j=1}^{n_i} p_{ij} x_{ij} \\ \text{przy ograniczeniach} \\ \sum_{i=1}^{k} \sum_{j=1}^{n_i} c_{ij} x_{ij} \leqslant b \\ x = (x_{ij}) \in X := \{(x_{ij}) \mid \sum_{j=1}^{n_i} x_{ij} = 1, \\ x_{ij} \in \{0,1\}, \ i = 1, ..., k, \ j = 1, ..., n_i\}. \end{array} \tag{1}$$

Zbiór X jest zbiorem wektorów zero-jedynkowych, podzielonych na k kategorii (odpowiadających zbiorom N_i, $i = 1, ...k$) i takich, że w każdej kategorii wszystkie współrzędne z wyjątkiem jednej są równe zero (warunek $\sum_{j=1}^{n_i} x_{ij} = 1$, $i = 1, ..., k$).

Problem $(MCKP)$ polega na wyborze elementu (elementów) ze zbioru X w taki sposób, aby całkowity koszt

$$\sum_{i=1}^{k} \sum_{j=1}^{n_i} c_{ij} x_{ij}$$

nie przekroczył danego $b \geqslant 0$ i aby zmaksymalizować całkowity zysk

$$\sum_{i=1}^{k} \sum_{j=1}^{n_i} p_{ij} x_{ij}.$$

Korzystając z powyższych notacji, problem $(MCKP)$ może być równoważnie zapisany w postaci wektorowej

$$(MCKP) \quad \begin{array}{l} \max \ p^T x \\ \text{przy ograniczeniach} \\ c^T x \leqslant b \\ x = (x_{ij}) \in X, \end{array}$$

gdzie p i c są wektorami z $\mathbb{R}^n$,

$$\begin{array}{l} p := (p_{11}, p_{12}, ..., p_{1n_1}, p_{21}, ..., p_{2n_2}, ..., p_{k1}, p_{k2}, ...p_{kn_k})^T \\ c := (c_{11}, c_{12}, ..., c_{1n_1}, c_{21}, ..., c_{2n_2}, ..., c_{k1}, c_{k2}, ...c_{kn_k})^T \end{array}$$

i dla dowolnych wektorów $u, v \in \mathbb{R}^n$, ich iloczyn skalarny $u^T v$ jest równy $\sum_{i=1}^{n} u_i v_i$.

Zbiór rozwiązań dopuszczalnych F problemu $(MCKP)$ jest definiowany za pomocą nierówności liniowej i ograniczenia $x \in X$, tzn.,

$$F := \{x \in \mathbb{R}^n \mid c^T x \leqslant b, \ \ x \in X\}$$

i ostatecznie

$$(MCKP) \quad \begin{array}{l} \max \ p^T x \\ \text{przy ograniczeniu} \\ x \in F. \end{array} \tag{2}$$

Optymalna wartość funkcji celu (profitu) problemu $(MCKP)$ jest równa $\max_{x\in F}\ p^T x$, zbiór rozwiązań S^* jest dany jako

$$S^* := \{\bar{x} \in F \mid p^T \bar{x} = \max_{x\in F}\ p^T x\}.$$

Ciągłą (liniową) relaksację problemu $(MCKP)$ definiujemy następująco:

$$\begin{array}{ll} & \max\ \ \sum_{i=1}^{k}\sum_{j=1}^{n_i} p_{ij}x_{ij} \\ & \text{przy ograniczeniach} \\ (C(MCKP)) & \sum_{i=1}^{k}\sum_{j=1}^{n_i} c_{ij}x_{ij} \leqslant b \\ & x = (x_{ij}) \in X := \{(x_{ij}) \mid \sum_{j=1}^{n_i} x_{ij} = 1, \\ & 0 \leqslant x_{ij} \leqslant 1,\ i = 1,...,k,\ j = 1,...,n_i\}. \end{array} \tag{3}$$

Rozwiązanie problemu $(C(MCKP))$ pozwala wyznaczyć oszacowanie od góry na wartość profitu rozwiązania problemu $(MCKP)$. Oszacowanie to będziemy oznaczać przez $UB(C(MCKP))$.

Aktualny stan wiedzy w zakresie algorytmów dla zagadnienia załadunku z wyborem i jego zastosowania

Problem $(MCKP)$ jest NP-trudny i nie może być rozwiązany w czasie wielomianowym [17]. Do rozwiązywania problemu $(MCKP)$ stosuje się najczęściej następujące podejścia:

a) preselekcja – stosowana do wstępnego przetwarzania danych problemu $(MCKP)$ celem redukcji rozmiaru problemu [17];

b) metoda *branch and bound*, w której dzieli się złożone zadania na wiele mniej złożonych/prostych zadań (*branches*) i usuwa się te, które nie mają wpływu na końcowe rozwiązanie (*bounding*) [8], [27], [33];

d) metody heurystyczne [1, 16];

e) programowanie dynamiczne [7], [29];

g) metody hybrydowe [9, 22, 32].

Istnieją algorytmy efektywnie rozwiązujące $(MCKP)$ bez sortowania i redukcji [10, 34] lub z sortowaniem i redukcją [6]. Oszacowania od góry na wartość optymalnego profitu mogą być uzyskane za pomocą tzw. relaksacji Lagrange'a. Wyżej wymienione podejścia i inne właściwości problemu $(MCKP)$ są opisane szczegółowo w monografiach [17, 23].

Dokładne algorytmy "branch-and-bound" [8] (programowanie całkowitoliczbowe), nawet te wykorzystujące komercyjne oprogramowanie optymalizacyjne (np. LINGO, CPLEX), mogą mieć problemy z rozwiązywaniem instancji problemu $(MCKP)$ wielkiej skali. Algorytm "branch-and-bound" z szybką metodą rozwiązywania liniowej relaksacji zredukowanego zadania, został zaproponowany przez Sinha and Zoltners [33]. Dudziński i Walukiewicz zaproponowali algorytm o pseudo-wielomianowej złożoności [7].

Algorytmy wykorzystujące programowanie dynamiczne wymagają całkowitoliczbowych wartości p_{ij}, c_{ij}, b i dla zadań wielkiej skali potrzebują wielkich zasobów pamięci do odtworzenia rozwiązania zadania (ang. *backtracking*) (znajdowanie rozwiązań w zbiorze F), patrz także monografia [23]. Proponowany w rozprawie algorytm nie wymaga,

by wartości p_{ij}, c_{ij}, b były liczbami całkowitymi oraz nie wymaga stosowania procedur typu 'backtracking'.

Algorytmy heurystyczne, bazujące na rozwiązywaniu problemu ($C(MCKP)$) i programowaniu dynamicznym [9, 29, 30, 14], są oceniane jako szybkie, ale mają ograniczenia typowe dla programowania dynamicznego.

Najnowsze podejście "reduce and solve" [5, 12] jest oparte na redukcji zadania poprzez zaproponowane pseudo cięcia, a następnie rozwiązaniu zredukowanych problemów za pomocą optymalizatora dla zadań programowania liniowego całkowitoliczbowego (ang. *Mixed Integer Programming*) – optymalizator MIP.

Wielowymiarowe zagadnienie załadunku z wyborem ($MMCKP$) i zagadnienie załadunku z wyborem ($MCKP$) są wariantami zagadnienia plecakowego (KP) i są stosowane do modelowania wielu problemów praktycznych, np. wybór puli projektów inwestycyjnych [27, 35], budżetowanie [30], planowanie przestrzeni reklamowej [33], dobór komponentów do budowy systemów IT [20, 31], zarządzanie sieciami komputerowymi [21], budowa adaptacyjnych systemów multimedialnych [18], kompresja plików [26], przydział przez stację bazową sub-częstotliwości dla obsługiwanych przez nią urządzeń mobilnych [3].

2 Zagadnienie optymalizacji wielokryterialnej

Niech $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., k,\ k > 1$, będą funkcjami zdefiniowanymi na $\mathbb{R}^n$ i $\Omega \subset \mathbb{R}^n$ będzie podzbiorem $\mathbb{R}^n$.

Problem optymalizacji wielokryterialnej jest definiowany, w najbardziej ogólnej postaci, jako

$$(P)\quad \begin{array}{l} \mathrm{V\,max}\quad f(x) := (f_1(x), ..., f_k(x)) \\ \text{przy ograniczeniu} \\ x \in \Omega, \end{array}$$

gdzie symbol $'\mathrm{V\,max}'$ oznacza operator wyznaczenia wszystkich rozwiązań Pareto optymalnych problemu (P), rozumianych w sensie Definicji 2.1.

Elementy zbioru Ω nazwiemy *rozwiązaniami*, a obrazy tych elementów w odwzorowaniu f – *ocenami rozwiązań* (lub po prostu – ocenami).

Niech

$$\mathbb{R}^k_+ := \{x = (x_1, ..., x_k) \in \mathbb{R}^k \ :\ x_i \geqslant 0,\ i = 1, ..., k\}.$$

Definicja 2.1 Punkt $x^* \in \Omega$ jest *Pareto optymalnym (Pareto efektywnym) rozwiązaniem* problemu (P), jeżeli

$$f(\Omega) \cap (f(x^*) + \mathbb{R}^k_+) = \{f(x^*)\}.$$

Innymi słowy, $x^* \in \Omega$ jest Pareto optymalnym rozwiązaniem problemu (P) jeśli nie istnieje $\bar{x} \in \Omega$ takie, że

$$\begin{array}{l} f_i(\bar{x}) \geqslant f_i(x^*) \text{ dla } \ i = 1, ..., k \ \text{ i} \\ f_\ell(\bar{x}) > f_\ell(x^*) \text{ dla pewnego } \ 1 \leqslant \ell \leqslant k. \end{array} \tag{4}$$

Zbiór wszystkich rozwiązań Pareto optymalnych zadania (P) nazwiemy *zbiorem Pareto*, a obraz tego zbioru w odwzorowaniu f – *frontem Pareto* zadania (P).

Obserwacja 2.1 Dwu-kryteriowy problem

$$\begin{array}{l} f_1(x) \to \max,\ f_2(x) \to \min \\ \text{przy ograniczeniu} \\ x \in \Omega \end{array}$$

z rozwiązaniami $x^* \in \Omega$ definiowanymi jako

$$(f_1(\Omega), f_2(\Omega)) \cap [(f_1(x^*), f_2(x^*)) + \mathbb{R}^2_{+-}] = \{(f_1(x^*), f_2(x^*))\} \tag{5}$$

gdzie

$$\mathbb{R}^2_{+-} := \{x = (x_1, x_2) \in \mathbb{R}^2 \ :\ x_1 \geqslant 0,\ x_2 \leqslant 0\}$$

jest równoważne problemowi

$$\begin{array}{l} f_1(x) \to \max,\ -f_2(x) \to \max \\ \text{przy ograniczeniu} \\ x \in \Omega \end{array}$$

w takim sensie, że ich zbiory Pareto są takie same (pokrywają się), a ich fronty Pareto różnią się jedynie znakiem drugiej współrzędnej.

Dwu-kryteriowy problem optymalizacyjny powiązany z $(MCKP)$

W powiązaniu z oryginalnym problemem $(MCKP)$ sformułowanym w (1), rozpatrywany jest następujący liniowy dwu-kryteriowy binarny problem optymalizacyjny $(BP1)$ postaci

$$\begin{array}{ll} & \sum_{i=1}^{k} \sum_{j=1}^{n_i} p_{ij} x_{ij} \to \max,\ \sum_{i=1}^{k} \sum_{j=1}^{n_i} c_{ij} x_{ij} \to \min \\ (BP1) & \text{przy ograniczeniu} \\ & (x_{ij}) \in X. \end{array} \tag{6}$$

W problemie (6), lewa strona liniowej nierówności ograniczenia $c^T x \leqslant b$, występującego w problemie $(MCKP)$, (1), staje się drugim kryterium i zbiór dopuszczalny zadania $(BP1)$ redukuje się do zbioru X.

Istnieją ważne powody do rozważenia dwu-kryteriowego problemu $(BP1)$. Pierwszy powód związany jest z faktem, że w problemie $(MCKP)$ nierówność

$$\sum_{i=1}^{k} \sum_{j=1}^{n_i} c_{ij} x_{ij} \leqslant b$$

jest postrzegana jako ograniczenie budżetu (w ogólności: zasobu), którego prawa strona jest nieprzekraczalną ilością zasobu. W dwu-kryteriowym problemie $(BP1)$ wymóg ten znika, a w zbiorze Pareto zadania $(BP1)$ mogą znajdować się rozwiązania, dla których ograniczenie budżetu nie jest spełnione. W Twierdzeniu 2.1, pokazane jest, że pod pewnymi, względnie łagodnymi warunkami, pośród rozwiązań dwu-kryteriowego problemu $(BP1)$ (lub równoważnie problemu (BP) – patrz dalej) występują rozwiązania problemu $(MCKP)$.

Drugi powód jest ważny z algorytmicznego punktu widzenia i jest powiązany z faktem, że w proponowanym w rozprawie algorytmie, można efektywnie wykorzystać specyficzną strukturę zbioru ograniczającego X, który zawiera k ograniczeń w postaci liniowych równań (każde odnoszące się do innej grupy zmiennych) i warunki binarności zmiennych.

Bardziej precyzyjnie, zbiór X może być reprezentowany przez iloczyn kartezjański

$$X = X^1 \times X^2 \times .... \times X^k, \tag{7}$$

zbiorów X^i, gdzie $X^i := \{x^i \in \mathbb{R}^{n_i} \mid \sum_{j=1}^{n_i} x_{ij} = 1, x_{ij} \in \{0,1\},\ j = 1, ..., n_i\}$, $i = 1, ..., k$ oraz

$$x = (\underbrace{x_{11}, ..., x_{1n_1}}_{x^1}, \underbrace{x_{21}, ..., x_{2n_2}}_{x^2}, ..., \underbrace{x_{k1}, ..., x_{kn_k}}_{x^k})^T, \tag{8}$$

tzn.,

$$x = (x^1, ..., x^k)^T$$

oraz $x^i = (x_{i1}, ...x_{in_i})$. Odpowiednio,

$$p = (p^1, ..., p^k)^T \ \text{ i } \ c = (c^1, ..., c^k)^T.$$

Warto zwrócić uwagę, że ze względu na ograniczenie budżetu, zbiór rozwiązań dopuszczalnych F problemu $(MCKP)$, (1), nie może być reprezentowany w formie analogicznej do (7).

Zgodnie z Obserwacją 2.1, problem $(BP1)$ może być równoważnie przeformułowane do postaci

$$(BP) \quad \begin{array}{l} \text{V}\max\ (p^T x,\ (-c)^T x) \\ \text{przy ograniczeniu} \\ x \in X. \end{array} \tag{9}$$

Relacje pomiędzy (BP) **i** $(MCPK)$

Problem $(MCKP)$ ma rozwiązanie, tzn. zbiór dopuszczalny F jest niepusty, jeżeli

$$b \geqslant \min_{x \in X} c^T x.$$

Z drugiej strony, jeżeli $b \geqslant \max_{x \in X} c^T x$, to zadanie $(MCKP)$ jest trywialne. Zatem, w konsekwencji zakładamy, że

$$C_{min} := \min_{x \in X} c^T x \leqslant b < C_{max} := \max_{x \in X} c^T x. \tag{10}$$

Niech $P_{max} := \max_{x \in X} p^T x$, tzn., P_{max} jest wartością maksymalną funkcji $p^T x$ na zbiorze X. Następujące obserwacje mają zasadnicze znaczenie dla dalszych rozważań. Możemy mieć do czynienia z dwoma przypadkami:

1. Pośród elementów zbioru X, które realizują wartość P_{max}, istnieje co najmniej jeden, który rozwiązuje $(MCKP)$, tzn. istnieje $x_p \in X$, $p^T x_p = P_{max}$ takie, że $c^T x_p \leqslant b$, tzn.

   $$C_{min} \leqslant c^T x_p \leqslant b < C_{max}. \tag{11}$$

   Zatem x_p rozwiązuje $(MCKP)$.

2. Żaden z elementów, które realizują maksymalną wartość P_{max}, nie jest rozwiązaniem $(MCKP)$, tzn. dla każdego $x_p \in X$, $p^T x_p = P_{max}$ mamy $c^T x_p > b$, tzn. żaden z elementów x_p realizujących wartość maksymalną P_{max} nie jest rozwiązaniem $(MCKP)$, tzn.

   $$C_{min} \leqslant b < c^T x_p \leqslant C_{max}. \tag{12}$$

W dalszej części koncentrujemy się na przypadku 2, scharakteryzowanym przez (12). Ten przypadek odnosi się do problemu (BP). Aby to zobaczyć wprowadzimy następującą notację. Niech $x_{cmin} \in X$ oraz $x_{pmax} \in X$ będą zdefiniowane jako

$$\begin{array}{lll} c^T x_{cmin} = C_{min} & \text{oraz} & p^T x_{cmin} = \max_{c^T x = C_{min}} p^T x \\ p^T x_{pmax} = P_{max} & \text{oraz} & c^T x_{pmax} = \min_{p^T x = P_{max}} c^T x. \end{array}$$

Niech S_{bo} będzie zbiorem Pareto problemu (BP),

$$S_{bo} := \{x \in X \ : \ (p^T, (-c)^T)(X) \cap [(p^T x, (-c)^T x) + \mathbb{R}^2_+] = \{(p^T x, (-c)^T x)\}\}. \qquad (13)$$

Zachodzi następujący lemat.

Lemat 2.1 *[2] Załóżmy, że zachodzi przypadek 2, tzn. spełniony jest warunek* (12)*. Istnieje rozwiązanie Pareto optymalne problemu* (BP)*,* $\bar{x} \in S_{bo}$*, które jest dopuszczalne dla problemu* $(MCKP)$*, tzn.* $c^T \bar{x} \leqslant b$*, co jest równoznaczne z faktem, że* $\bar{x} \in F$*.*

Można zatem sformułować relację pomiędzy rozwiązaniami problemu $(MCKP)$ i Pareto optymalnymi rozwiązaniami problemu (BP), w przypadku, gdy spełniony jest warunek (12).

Twierdzenie 2.1 *[2] Załóżmy, że mamy problem* $(MCKP)$*, spełniający warunek* (12)*. Niech* $x^* \in X$ *będzie rozwiązaniem Pareto optymalnym problemu* (BP)*, takim, że*

$$b - c^T x^* = \min_{x \in S_{bo}, b - c^T x \geqslant 0} b - c^T x. \qquad (*) \qquad (14)$$

Zatem x^* *rozwiązuje* $(MCKP)$*.*

Twierdzenie 2.1 mówi, że pod warunkiem spełnienia (12) każde rozwiązanie problemu (BP) spełniające warunek $(*)$ (14) rozwiązuje problem $(MCKP)$. Ogólny związek pomiędzy skalarną optymalizacją z ograniczeniami a optymalizacją wielokryterialną jest badany, np. [19]. Twierdzenie 2.1 zilustrowano na rysunku 1.

Bazując na Twierdzeniu 2.1, skonstruowano algorytm znajdujący rozwiązanie Pareto optymalne $x \in S_{bo}$ problemu (BP), które jest dopuszczalne dla problemu $(MCKP)$ i dla którego warunek $(*)$ (14) jest spełniony, albo w pewnym sensie, jest to rozwiązanie, które w najmniejszym stopniu łamie ten warunek.

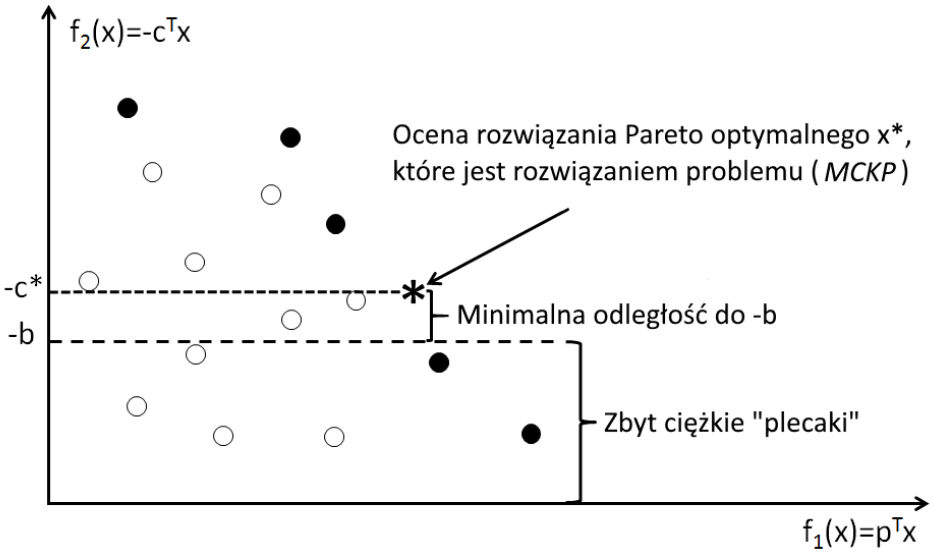
3 Skalaryzacja problemu (BP) i jego rozwiązanie

Dla problemu (BP) zachodzi następujące, klasyczne twierdzenie o skalaryzacji.

Twierdzenie 3.1 *[11, 25] Jeżeli istnieje* $\lambda_\ell > 0$*,* $\ell = 1, 2$*, takie, że* $x^* \in X$ *jest rozwiązaniem dla zeskalaryzowanego problemu*

$$(BS(\lambda_1, \lambda_2)) \quad \max_{x \in X} \ \lambda_1 p^T x + \lambda_2 (-c)^T x \qquad (15)$$

to x^* *jest Pareto optymalnym rozwiązaniem problemu* (BP)*.*



Rys. 1: Ilustracja do Twierdzenia 2.1; ● – oceny rozwiązań Pareto optymalnych problemu (BP), $*$ – ocena Pareto optymalnego rozwiązania problemu (BP) i jednocześnie rozwiązanie problemu $(MCKP)$.

Powyższe Twierdzenie pozostaje prawdziwe przy normalizacji wektora λ, tzn. przy warunku $\sum_{l=1}^{2} \lambda_\ell = 1$. W związku z tym w dalszej części, dla $0 < \lambda < 1$, rozpatrywane jest zeskalaryzowane zadanie postaci

$$(BS(\lambda)) \quad \max_{x \in X} \lambda p^T x + (1-\lambda)(-c)^T x \tag{16}$$

Obserwacja 3.1 Zgodnie z Twierdzeniem 3.1, rozwiązania problemów

$$\max_{x \in X} p^T x, \quad \max_{x \in X} (-c)^T x \tag{17}$$

nie muszą być Pareto optymalne, ponieważ wagi nie są obie dodatnie. Jednak rozwiązania Pareto optymalne v, w problemu (BP), dla których $p^T v = max_{x \in S_{bo}} p^T x$, $(-c)^T w = max_{x \in S_{bo}} (-c)^T x$, mogą być znalezione w inny sposób. Mianowicie, istnieją $\varepsilon_1 > 0$ i $\varepsilon_2 > 0$, takie, że rozwiązania problemów

$$(P1) \quad \max_{x \in X} p^T x + \varepsilon_1 (-c)^T x \tag{18}$$

oraz

$$(P2) \quad \max_{x \in X} (-c)^T x + \varepsilon_2 p^T x \tag{19}$$

są odpowiednimi Pareto optymalnymi rozwiązaniami problemów (17). Współczynniki ε_1 i ε_2 są określone w (27).

Wartość funkcji celu rozwiązania skalarnego problemu optymalizacyjnego będziemy dalej nazywać *wartością optymalną* tego problemu.

Dekompozycja

Ze względu na ważne właściwości strukturalne zbioru X i możliwość zapisania X w postaci (7),

$$X = X^1 \times X^2 \times .... \times X^k,$$

można przedstawić formuły na rozwiązanie problemu $(BS(\lambda))$ (16) dane w postaci jawnych wyrażeń . W tym celu dekomponuje się problemy $(BS(\lambda))$ w następujący sposób.

Za pomocą (8) można zapisać dowolny $x \in X$ w następującej postaci

$$x := (x^1, x^2, ..., x^k)^T,$$

gdzie $x^i = (x_{i1}, ..., x_{in_i})$, $i = 1, ..., k$, oraz $\sum_{j=1}^{n_i} x_{ij} = 1$.

Niech $0 < \lambda < 1$. Zgodnie z (7) mamy

$$X^i := \{x^i = (x_{i1}, ..., x_{in_i}) \in \mathbb{R}^{n_i} \ : \ \sum_{j=1}^{n_i} x_{ij} = 1, \quad x_{ij} \in \{0, 1\}, \quad j = 1, ..., n_i\}$$

dla $i = 1, ..., k$. Rozważmy problemy $(BS(\lambda))_i$, $i = 1, ..., k$, postaci

$$(BS(\lambda))_i \quad \max_{x^i \in X^i} \lambda (p^i)^T x^i + (1 - \lambda)(-c^i)^T x^i \tag{20}$$

Rozwiązując problemy $(BS(\lambda))_i$, $i = 1, ..., k$, znajdujemy ich rozwiązania $\bar{x}^i \in \mathbb{R}^{n_i}$. Wektor

$$\bar{x} := (\bar{x}^1, ..., \bar{x}^k)^T$$

rozwiązuje $(BS(\lambda))$. Zatem problem $(BS(\lambda))$ (16) jest zdekomponowany do k podproblemów $(BS(\lambda))_i$ (20), rozwiązania których tworzą rozwiązanie problemu $(BS(\lambda))$ (16).

Poniżej podano formuły znajdowania rozwiązań problemów $(BS(\lambda))_i$ (20). Dla $i = 1, .., k$ niech

$$V_i := \max\{\lambda p_{ij} + (1 - \lambda)(-c_{ij}) \ : \ 1 \leqslant j \leqslant n_i\} \tag{21}$$

i niech $1 \leqslant j_i^* \leqslant n_i$ będzie numerem indeksu, dla którego jest osiągana wartość V_i, tzn.

$$V_i = \lambda p_{ij_i^*} + (1 - \lambda)(-c_{ij_i^*}). \tag{22}$$

Numer indeksu j_i^*

$$\bar{x}^i := (0, .., \underbrace{1}_{j_i^*}, 0, ..., 0)^T \tag{23}$$

jest rozwiązaniem $(BS(\lambda))_i$, natomiast

$$\bar{x}^* := (\bar{x}^1, \bar{x}^2, ..., \bar{x}^k)^T \tag{24}$$

jest rozwiązaniem $(BS(\lambda))$ (16). Optymalną wartość $(BS(\lambda))$ otrzymujemy następująco

$$V := V_1 + ... + V_k. \tag{25}$$

W ten sposób wykazaliśmy fakt następujący:

Propozycja 3.1 *Każdy element $\bar{x}^i \in \mathbb{R}^{n_i}$ dany przez* (23) *jest rozwiązaniem problemu $(BS(\lambda))_i$ dla $i = 1, ..., k$, oraz każdy $\bar{x}^* \in \mathbb{R}^n$, dany przez* (24)*, jest rozwiązaniem problemu $(BS(\lambda))$.*

Zauważmy, że każdy problem optymalizacyjny $(BS(\lambda))_i$ może być rozwiązany w czasie $O(n_i)$ (koszt znajdowania maksymalnej liczby w zadanym zbiorze liczb), zatem problem $(BS(\lambda))$ może być rozwiązany w czasie $O(n)$, gdzie $n = \sum_{i=1}^{k} n_i$.

Może istnieć więcej niż jedno rozwiązanie problemu $(BS(\lambda))_i$, $i = 1, ..., k$. Dla potrzeb rozwiązania problemu $(MCKP)$, zgodnie z Twierdzeniem 2.1, spośród wszystkich rozwiązań $(BS(\lambda))$ wybieramy to, dla którego wartość drugiego kryterium jest większa niż lub równa $-b$ i jednocześnie różnica wartości drugiego kryterium i $-b$ jest najmniejsza (gdzie b jest prawą stroną ograniczenia budżetu, por. rys. 1).

Dla potrzeb dalszych rozważań założymy, że wektor p ma co najmniej dwie różne współrzędne oraz wektor c ma co najmniej dwie różne współrzędne.

Stosując Propozycję 3.1, można łatwo rozwiązać problemy $(P1)$ i $(P2)$ zdefiniowane w Obserwacji 3.1, tzn. poprzez zastosowanie (25) natychmiast otrzymujemy

$$F_1 := \max_{x \in X} p^T x, \quad F_2 := \max_{x \in X} (-c)^T x,$$

które są wartościami optymalnymi, odpowiednio, $(P1)$ i $(P2)$, a poprzez zastosowanie formuły (24) znajdujemy ich odpowiednie rozwiązania $\bar{x}_1$ i $\bar{x}_2$.

Zgodnie z Obserwacją 3.1, propozycja 3.1 i formuła (24) pozwalają znaleźć $\varepsilon_1 > 0$ oraz $\varepsilon_2 > 0$. Za pomocą (24), łatwo można znaleźć elementy $\bar{x}_1, \bar{x}_2 \in X$, takie że

$$F_1 = p^T \bar{x}_1, \quad F_2 = (-c)^T \bar{x}_2.$$

Podstawiając

$$\bar{F}_1 := p^T \bar{x}_2, \quad \bar{F}_2 := (-c)^T \bar{x}_1$$

i niech

$$\bar{V}_1 := F_1 - decr(p), \quad \bar{V}_2 := F_2 - decr(-c),$$

gdzie $decr(p)$ i $decr(-c)$ oznaczają najmniejszy niezerowy spadek wartości funkcji $p^T x$ i $(-c)^T x$ odpowiednio z F_1 i F_2. Wartości $decr(p)$ i $decr(-c)$ mogą być łatwo znalezione.

Obserwacja 3.2 Następujące formuły opisują $decr(p)$ oraz $decr(-c)$,

$$decr(p) := \min_{1 \leqslant i \leqslant k} (p^i_{max} - p^i_{submax}), \quad decr(-c) := \min_{1 \leqslant i \leqslant k} ((-c)^i_{max} - (-c)^i_{submax}), \tag{26}$$

gdzie p^i_{max} oraz $(-c)^i_{max}$, $i = 1, ..., k$, są zdefiniowane poprzez (8), p^i_{submax}, $(-c)^i_{submax}$, $i = 1, ..., k$, są submaksymalnymi wartościami funkcji $(p^i)^T x^i$, $((-c)^i)^T x^i$, $x^i \in X^i$, $i = 1, ..., k$.

Dla dowolnego $1 \leqslant i \leqslant k$, submaksymalne wartości dowolnej funkcji liniowej $d^T x^i$ na X^i, gdzie wektor $d \in \mathbb{R}^{n_i}$ ma co najmniej dwie różne współrzędne, mogą być znalezione przez: uporządkowanie nierosnące współczynników wektora d,

$$d_{j1} \geqslant d_{j2} \geqslant ... \geqslant d_{jn_i},$$

a następnie obserwując, że submaksymalna (tzn. mniejsza niż maksymalna ale możliwie najbliższa do maksymalnej) wartość $d^T x^i$ na X^i jest osiągana dla

$$\bar{x}^i := (0, ..., \underbrace{1}_{j2}, 0, ...0).$$

W przypadku, gdy wektor d ma więcej niż jedną współrzędną o wartości maksymalnej w zbiorze wartości tych współrzędnych, znalezienie współrzędnej tego wektora o wartości submaksymalnej, również bazuje na nierosnącym uporządkowaniu współrzędnych wektora d.

Na podstawie Obserwacji 3.2 można znaleźć wartości p^i_{submax} oraz $(-c)^i_{submax}$ w czasie $O(n_i)$, $i = 1, ..., k$. Może to być wykonane dla danego i przez znalezienie maksymalnej wartości pośród wszystkich p_{ij} $(-c_{ij})$, $j = 1, ..., n_i$, oprócz p^i_{max} $(-c^i_{max})$. Dlatego koszt obliczeniowy znalezienia $decr(p)$ i $decr(-c)$ wynosi $O(n)$.

Mamy następujący fakt.

Propozycja 3.2 *Niech F_1, F_2, $\bar{F}_1$, $\bar{F}_2$, $\bar{V}_1$, $\bar{V}_2$ będą zdefiniowane jak powyżej. Problemy*

$$(P1) \quad \max_{x \in X} p^T x + \varepsilon_1 (-c)^T x$$

oraz

$$(P2) \quad \max_{x \in X} (-c)^T x + \varepsilon_2 p^T x$$

gdzie

$$\varepsilon_1 := \frac{F_1 - \bar{V}_1}{F_2 - \bar{F}_2}, \quad \varepsilon_2 := \frac{F_2 - \bar{V}_2}{F_1 - \bar{F}_1}, \tag{27}$$

dają Pareto optymalne rozwiązania problemu (BP), oraz odpowiednio $\bar{x}_1$ i $\bar{x}_2$. Ponadto,

$$f_1(\bar{x}_1) = F_1 \quad i \quad f_2(\bar{x}_2) = F_2,$$

tzn. $\bar{x}_1$, $\bar{x}_2$ rozwiązują, odpowiednio, problemy (17).

Dowód. Wynika bezpośrednio z przyjętych zapisów, patrz rys. 2. Przykładowo, funkcja celu problemu $(P1)$ jest reprezentowana przez linię prostą przechodzącą przez punkty $(F_1, \bar{F}_2)$ i $(\bar{V}_1, F_2)$, tzn.

$$F_1 + \varepsilon_1 \bar{F}_2 = \bar{V}_1 + \varepsilon_1 F_2,$$

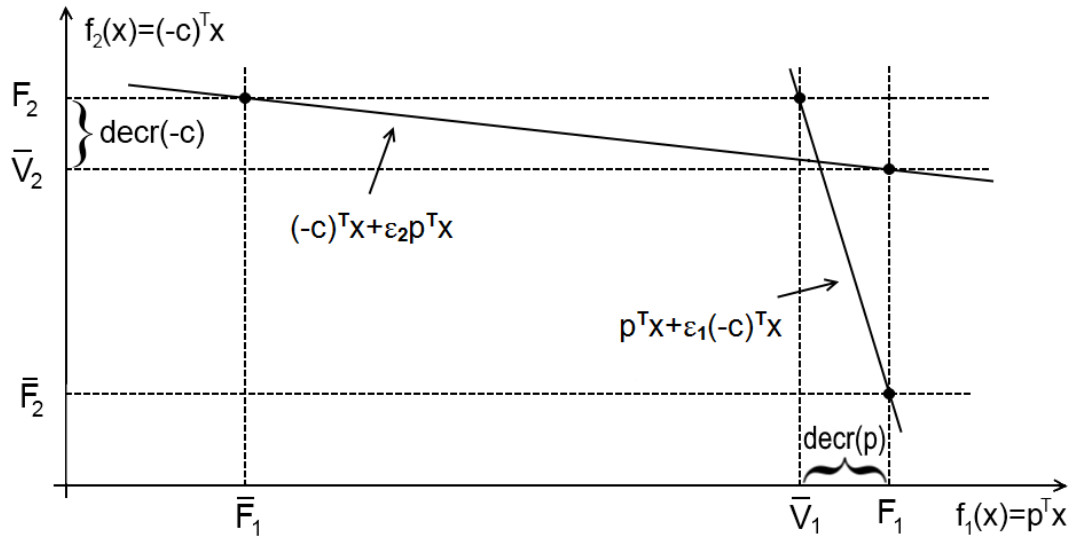
które dają (27). Wybór F_1, $\bar{F}_2$ i $\bar{V}_1$, F_2 gwarantuje, że $\bar{x}_1$ rozwiązuje problem $(P1)$ (i analogicznie dla $\bar{x}_2$, który rozwiązuje problem $(P2)$). ■

4 Nowe algorytmy dla zagadnienia załadunku z wyborem

Algorytm BISSA znajdowania rozwiązania przybliżonego

Proponowany algorytm BISSA (ang. *Bi-objective Approximate Solution Search Algorithm*) przeszukuje zbiór Pareto problemu (BP), w celu znalezienia elementu $\hat{x} \in F$, który jest (w ogólnym przypadku) przybliżonym rozwiązaniem problemu $(MCKP)$. Istotą działania algorytmu jest rozwiązywanie sekwencji problemów $(BS(\lambda))$ zdefiniowanych przez (16) dla $0 < \lambda < 1$, określonych w taki sposób, że rozwiązania Pareto optymalne $x(\lambda)$ problemu $(BS(\lambda))$ są dopuszczalne dla $(MCKP)$ i dla których $(-c)^T x(\lambda) + b \geqslant 0$ i $(-c)^T x(\lambda) + b$ zmniejsza się dla kolejnych λ.

Zgodnie z Twierdzeniem 3.1, każde rozwiązanie $(BS(\lambda))$ jest pewnym rozwiązaniem Pareto optymalnym problemu (BP). Zgodnie z Twierdzeniem 2.1, każde Pareto optymalne



Rys. 2: Konstrukcja ε_1 i ε_2.

rozwiązanie x^* problemu (BP), które jest dopuszczalne dla $(MCKP)$, tzn. $(-c)^T x^* \geqslant -b$ i spełniony jest warunek $(*)$ (14), tzn.

$$(-c)^T x^* + b = \min_{x \in S_{bo}, (-c)^T x + b \geqslant 0} (-c)^T x + b, \qquad (*)$$

rozwiązuje problem $(MCKP)$. Ponieważ problem $(BS(\lambda))$ (16) jest skalaryzacją liniową problemu (BP), nie jesteśmy w stanie, w ogólności, wyznaczyć wszystkich $x \in S_{bo}$ (13), takich, że $(-c)^T x + b \geqslant 0$, aby znaleźć x^*, który spełnia warunek $(*)$, gdzie S_{bo} oznacza zbiór wszystkich rozwiązań Pareto problemu dwu-kryteriowego (BP). Z drugiej strony, używając skalaryzacji liniowej i korzystając z Propozycji 3.1, jesteśmy w stanie zdekomponować i łatwo rozwiązać problem $(BS(\lambda))$.

W oparciu o Twierdzenie 2.1 algorytm `BISSA` poszukuje Pareto optymalnego rozwiązania $\hat{x} \in X$ problemu (BP), które jest dopuszczalne dla $(MCKP)$, tzn. $c^T \hat{x} \leqslant b$, dla którego wartość $b - c^T \hat{x}$ jest najmniejsza możliwa (ale niekoniecznie minimalna) i zbliża się do warunku $(*)$ Twierdzenia 2.1 najbliżej, jak to jest możliwe przy wykorzystaniu skalaryzacji liniowej do wyznaczania rozwiązań Pareto optymalnych problemu (BP).

Opis działania algorytmu `BISSA`

1. Znalezienie rozwiązań problemów $(P1)$ (18) i $(P2)$ (19), jak również ich ocen (linie 1 – 5). Są to tzw. skrajne rozwiązania Pareto optymalne problemu (BP) (Propozycja (3.2)). Te oceny (punkty), nazwane $(a_1, b_1)^0$ oraz $(a_2, b_2)^0$, są przedstawione na rys. 3.

2. Porównanie wartości drugich współrzędnych znalezionych punktów z wartością $-b$, w celu sprawdzenia, czy istnieje rozwiązanie problemu $(MCKP)$ (linie 6 – 9). Jeżeli do tego momentu (linia 10) rozwiązanie nie zostało znalezione, można rozpocząć przeszukiwanie przestrzeni $\mathbb{R}^2$ (a więc i zbioru X). Decyzje o kierunku poszukiwań (zawężaniu $\mathbb{R}^2$) podejmujemy obserwując wartości $f(x) \in \mathbb{R}^2,\ x \in X$.

3. Obliczenie wartości λ (linia 13), która określa nachylenie linii prostej, łączącej (a_1, b_1) i (a_2, b_2). Jednocześnie jest to parametr skalaryzacyjny definiujący pro-

Algorytm 1 BISSA – Bi-objective Approximate Solution Search Algorithm

Wyznacz ε_1, ε_2 za pomocą (27)
Oznaczmy $f_1(x) = p^T x$ and $f_2(x) = (-c)^T x$
Rozwiąż $(P1)$ za pomocą 3.2 ▷ x_1 jest rozwiązaniem $(P1)$
Rozwiąż $(P2)$ za pomocą 3.2 ▷ x_2 jest rozwiązaniem $(P2)$
$a_1 := f_1(x_1)$, $b_1 := f_2(x_1)$, $a_2 := f_1(x_2)$, $b_2 := f_2(x_2)$
if $(a_1, b_1) = (a_2, b_2)$ and $b_2 \geqslant -b$ **then** x_2 jest rozwiązaniem $(MCKP)$ and STOP **end if**
if $b_1 \geqslant -b$ **then** x_1 jest rozwiązaniem $(MCKP)$ and STOP **end if**
if $b_2 = -b$ **then** x_2 jest rozwiązaniem $(MCKP)$ and STOP **end if**
if $b_2 < -b$ **then** brak rozwiązania problemu $(MCKP)$ and STOP **end if**
▷ $(a_1, b_1) \neq (a_2, b_2)$ oraz $b_1 < -b < b_2$. Przegląd przestrzeni poszukiwań
$loop := TRUE$
repeat
    $\lambda := \frac{(b_2 - b_1)}{(a_1 - a_2) + (b_2 - b_1)}$, $\alpha := \lambda a_1 + (1 - \lambda) b_1$ ▷ $0 < \lambda < 1$
    Rozwiąż $(BS(\lambda))$ za pomocą (24) and (25) ▷ x rozwiązanie, *opt* wartość optymalna, S zbiór rozwiązań zadania $(BS(\lambda))$
    if $opt > \alpha$ **then**
        if $f_2(x) > -b$ **then**
            $a_2 := f_1(x)$, $b_2 := f_2(x)$
        else if $f_2(x) < -b$ **then**
            $a_1 := f_1(x)$, $b_1 := f_2(x)$
        else
            x jest rozwiązaniem $(MCKP)$ and STOP
        end if
    else ▷ $opt = \alpha$
        $\hat{x} := \arg\min_{x \in S, f_2(x) \geqslant -b} f_2(x)$
        $loop := FALSE$
    end if
until $\neg loop$ ▷ $\hat{x}$ przybliżone rozwiązanie zadania $(MCKP)$ ▷ $f_1(\hat{x})$ oszacowanie od dołu na wartość optymalnego profitu (LB) zadania $(MCKP)$
$u := \frac{(a_1 - f_1(\hat{x}))(f_2(\hat{x}) + b)}{f_2(\hat{x}) - b_1}$ ▷ $f_1(\hat{x}) + u$ oszacowanie od góry na wartość optymalnego profitu (UB) zadania $(MCKP)$

blem ($BS(\lambda)$) (formuła (16)). Ocena rozwiązania problemu ($BS(\lambda)$) nie może leżeć poniżej linii prostej określonej punktami (a_1, b_1) i (a_2, b_2). Musi ona znajdować się na tej linii lub powyżej, ponieważ jest to Pareto optymalne rozwiązanie problemu (BP).

4. Rozwiązanie problemu ($BS(\lambda)$) za pomocą formuł (23) i (24) (linia 14).

5. Wykonywanie w pętli (linie 15 – 27) "skanowania" przestrzeni poszukiwań w celu znalezienia rozwiązań Pareto optymalnych problemu (BP), które są dopuszczalne dla problemu ($MCKP$).

   - Jeżeli istnieją oceny rozwiązań leżące nad linią prostą, określoną przez λ (warunek w linii 15 jest spełniony), to albo następuje zawężenie obszaru poszukiwań (poprzez określenie nowych punktów (a_1, b_1) i (a_2, b_2) z górnym indeksem równym 1 (patrz rys. 3) i pętla dalej się wykonuje, albo uzyskiwane jest rozwiązanie problemu ($MCKP$) i pętla jest przerywana.
   - Jeżeli nie istnieją oceny rozwiązań leżące nad linią prostą, określoną przez λ (warunek w linii 15 nie jest spełniony), to rozwiązanie x ze zbioru S (zbiór rozwiązań problemu ($BS(\lambda)$)) z wartością leżącą powyżej linii prostej, wyznaczonej przez $-b$ (rozwiązanie dopuszczalne dla problemu ($MCKP$)), dla którego wartość $f_2(x) + b$ jest minimalna w tym zbiorze, jest przybliżonym rozwiązaniem ($\hat{x}$) problemu ($MCKP$) i pętla jest przerywana.

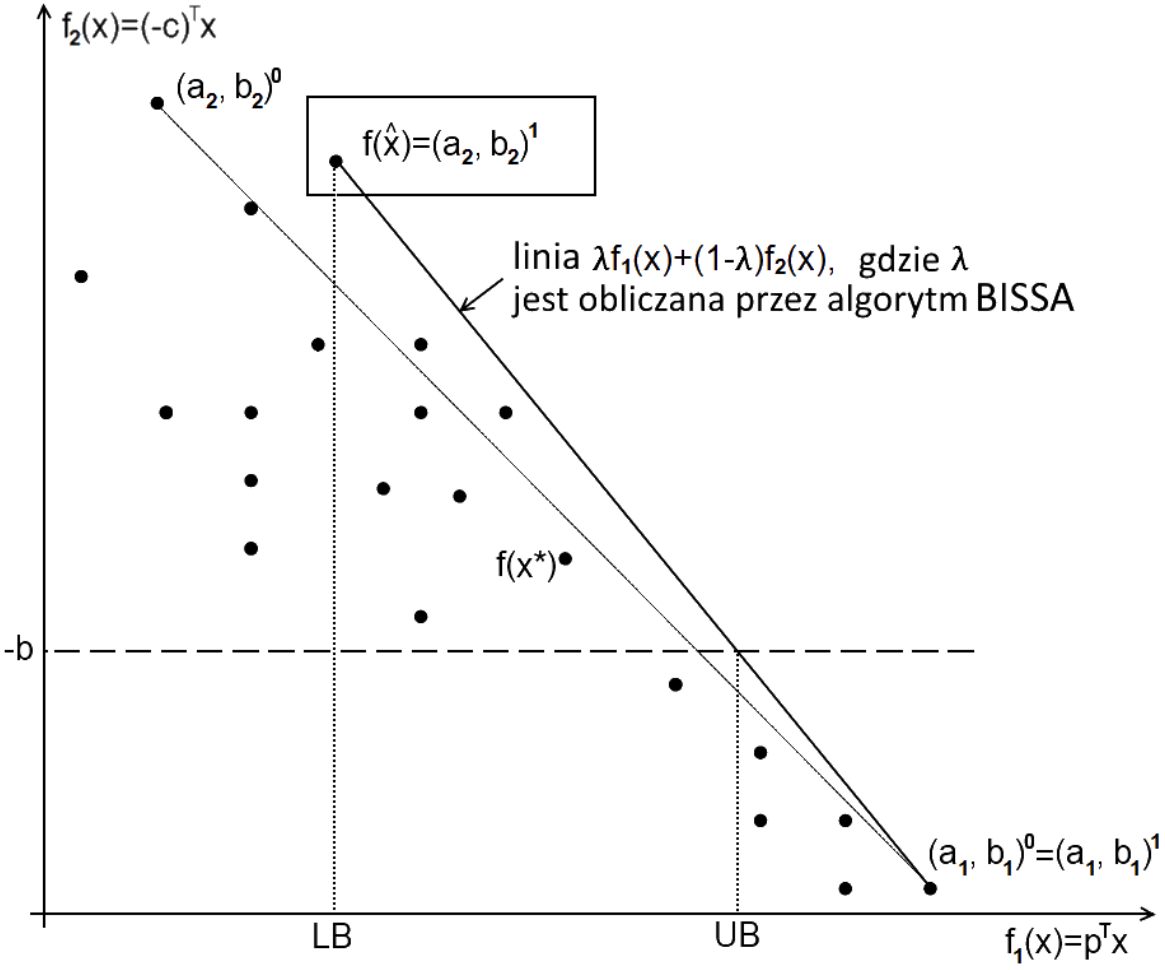
6. Obliczenie oszacowania od góry $f_1(\hat{x})+u$ na wartość profitu dokładnego rozwiązania problemu ($MCKP$) (linia 28).

Algorytm `BISSA` znajduje dokładne rozwiązanie problemu ($MCKP$), albo rozwiązanie przybliżone $\hat{x}$ tego problemu (które dostarcza oszacowania (LB) od dołu na wartość optymalną tego problemu) oraz oszacowanie od góry (UB) na wartość optymalną tego problemu (patrz rys. 3). Rozwiązanie znajdowane przez algorytm `BISSA` jest, w ogólności, tylko przybliżonym rozwiązaniem problemu ($MCKP$), ponieważ trójkąt określony punktami $(f_1(\hat{x}), f_2(\hat{x})), (f_1(\hat{x})+u, -b), (f_1(\hat{x}), -b)$ (zwany dalej 'trójkątem niepewności') może zawierać inne oceny Pareto (kandydatów na oceny dokładnych rozwiązań problemu ($MCKP$)), których proponowany algorytm `BISSA` nie jest w stanie dostarczyć. Powodem jest wykorzystanie techniki skalaryzacyjnej, opartej na ważonej sumie funkcji celów, w celu znalezienia rozwiązań Pareto problemu (BP).

Każda instancja problemu optymalizacyjnego ($BS(\lambda)$) (16) może być rozwiązana w czasie $O(n)$, ale liczba tych instancji rozwiązywanych przez proponowany algorytm zależy od rozmiaru problemu (wartości k oraz n_i), a także od danych.

Zaproponowany w rozprawie algorytm `BISSA` dostarcza, w ogólnym wypadku, rozwiązanie dopuszczalne problemu ($MCKP$), będące rozwiązaniem przybliżonym problemu ($MCKP$), poprzez rozwiązanie pewnej liczby (zwykle bardzo małej) zadań ($BS(\lambda)$). Główną zaletą oparcia algorytmu przybliżonego `BISSA` na rozwiązywaniu problemu (*de facto*, serii problemów) ($BS(\lambda)$) (16) jest to, że jego rozwiązanie zadane jest przez relatywnie proste, jawne wyrażenia (21) – (24).

Wyniki działania algorytmu przybliżonego `BISSA` mogą być dobrym punktem wyjścia do dalszych poszukiwań lepszego rozwiązania problemu ($MCKP$) za pomocą innych algorytmów (np. heurystycznych, dokładnych).



Rys. 3: Ocena $f(\hat{x})$, gdzie $\hat{x}$ jest rozwiązaniem przybliżonym problemu $(MCKP)$, znalezionym przez algorytm `BISSA`, LB i UB to, odpowiednio, oszacowania od dołu i od góry na wartość optymalną problemu $(MCKP)$, x^* – rozwiązanie problemu $(MCKP)$.

Algorytm dokładny SKAN-DOK

W rozprawie zaprezentowano algorytm dokładny SKAN-DOK znajdowania rozwiązania dokładnego. Ma on przede wszystkim charakter teoretyczny. Ze względu na wykładniczy wzrost rozmiarów macierzy wykorzystywanej do obliczeń, stosowanie tego algorytmu jest możliwe jedynie dla problemów niewielkich rozmiarów.

Algorytm heurystyczny SKAN-LIDER

W rozprawie zaprezentowano algorytm heurystyczny SKAN-LIDER, próbujący znaleźć lepsze rozwiązanie problemu ($MCKP$) niż to, dostarczone przez algorytm BISSA. Parametrem algorytmu $lg \in \{1, ..., k\}$ steruje się liczbą kategorii, w których podmienia się odpowiednie elementy (tzw. "wiceliderów") w rozwiązaniu $\hat{x}$. Przez podmianę rozumiemy tu utworzenie innego wektora binarnego, należącego do zbioru dopuszczalnego zadania ($MCKP$). Za pomocą parametru lg można sterować możliwościami wyszukiwania algorytmu. Im wyższa wartość tego parametru, tym większa liczba sprawdzanych rozwiązań. Oceny wszystkich rozwiązań znajdują się na liniach prostych, równoległych do prostej określonej przez λ^* (i leżących pod tą prostą), ale do poprawienia wyniku algorytmu BISSA uwzględnia się tylko te rozwiązania, których oceny znajdują się w trójkącie niepewności.

Część II
Eksperyment obliczeniowy

5 Testy algorytmu BISSA

W dostępnej literaturze przedmiotu nie stwierdzono zadań testowych (benchmarków) dla problemu ($MCKP$). Do generowania danych testowych został użyty program [28] autorstwa Davida Pisingera. Program ten generuje zadanie ($MCKP$) z losowymi danymi, a następnie oblicza, zgodnie z algorytmem EXACT [29], wartość optymalną tego zadania – profit optymalny. Autorska modyfikacja programu [28] pozwalała na zapis do plików tekstowych danych wygenerowanego zadania ($MCKP$). Dodatkowo program [28] wyznacza za pomocą algorytmu zachłannego GREEDY [29], rozwiązanie problemu $C(MCKP)$. Dostarcza on tym samym wartość $UB(C(MCKP))$. Program [28] nie dostarcza rozwiązań zadań ($MCKP$), a więc numerów indeksów przedmiotów w kategoriach, dających rozwiązanie optymalne.

Ponieważ trudność zadań plecakowych zależy od stopnia korelacji pomiędzy profitem i kosztem przedmiotów (elementów) (patrz np. monografia [23]), przeprowadzono dwa testy numeryczne: Eksperyment 1 z nieskorelowanymi (unc) instancjami danych (łatwe do rozwiązania) i Eksperyment 2 ze słabo skorelowanymi (wco) instancjami danych (trudniejsze do rozwiązania) (cf. [15]).

W Eksperymencie 1, dla każdego problemu testowego, profity (p_{ij}) i koszty (c_{ij}) elementów były losowo generowanymi liczbami całkowitymi [1] (zgodnie z rozkładem jedno-

[1] Wartości profitów i kosztów były liczbami całkowitymi, ze względu na wykorzystanie do porównań

stajnym) z przedziału $[1, R]$. Dla każdego problemu testowego całkowity koszt b był równy $c + random(0, \frac{1}{4} * c)$ albo $c - random(0, \frac{1}{4} * c)$, i był losowo wybrany, z takim samym prawdopodobieństwem równym $0,5$, gdzie $c = \frac{1}{2}\sum_{i=1}^{k}(\min_{j=1,\ldots,n_i} c_{ij} + \max_{j=1,\ldots,n_i} c_{ij})$, $random(0, r)$ oznacza losowo wybraną (zgodnie z rozkładem jednostajnym) liczbę całkowitą z przedziału $[0, r]$.

Wygenerowano 6 instancji danych (unc): $(unc, 10, 1000)$, $(unc, 1000, 10)$, $(unc, 100, 100)$, $(unc, 100, 1000)$, $(unc, 1000, 100)$, $(unc, 1000, 1000)$. Przykładowe wyniki dla problemów $(unc, 10, 1000)$, tj. profit wyznaczonych przez testowane algorytmy rozwiązań i liczba rozwiązanych problemów ($BS(\lambda)$) dla każdego problemu, są przedstawione w tabeli (1).

Nagłówki kolumn w tabelach (1), (3) oznaczają:

1 – id problemu,

2 – profit dokładnego rozwiązania znaleziony przez algorytm `EXACT`,

3 – profit przybliżonego rozwiązania (LB) znaleziony przez algorytm `BISSA`,

4 – różnica pomiędzy wartościami w kolumnach 2 i 3,

5 – względna różnica (%) pomiędzy wartościami w kolumnach 2 i 3 ((`EXACT`-(LB))/(LB))*100%,

6 – wartość $UB(C(MCKP))$ wyznaczona przez algorytm `GREEDY`,

7 – wartość (UB) znaleziona przez algorytm `BISSA`,

8 – różnica pomiędzy wartościami w kolumnach 7 i 3 (UB)-(LB),

9 – względna różnica (%) pomiędzy wartościami w kolumnach 7 i 3 (((UB)-(LB))/(LB))*100%,

10 – liczba problemów ($BS(\lambda)$) rozwiązywanych przez algorytm `BISSA` (bez $P1$ i $P2$).

W Eksperymencie 2, dla każdego problemu testowego, koszty elementów (c_{ij}) w zbiorze N_i były losowo generowanymi liczbami całkowitymi (zgodnie z rozkładem jednostajnym) z przedziału $[1, R]$. Profity elementów (p_{ij}) w tym zbiorze były generowane losowo (zgodnie z rozkładem jednostajnym) w zakresie $[c_{ij} - 10, c_{ij} + 10]$, tak że $p_{ij} \geqslant 1$. Dla każdego problemu testowego całkowity koszt b był obliczany tak, jak w Eksperymencie 1.

Wygenerowano 2 instancje danych (wco): $(wco, 10, 10)$, $(wco, 20, 20)$. Przykładowe wyniki dla problemów $(wco, 10, 10)$ są przedstawione w tabeli (3).

Indeksy przedmiotów w poszczególnych kategoriach dla problemów $(unc, 10, 1000)$ i $(wco, 10, 10)$, znalezione przez algorytm `BISSA`, są przedstawione w tabelach, odpowiednio, (2) i (4).

5.1 Testy algorytmu heurystycznego `SKAN-LIDER`

Program z implementacją algorytmu `SKAN-LIDER` odczytywał pliki z wynikami obliczeń algorytmu `BISSA` i wykonywał zupełny przegląd kombinacji (od 1 do zadanej liczności $lgrup$-elementowych) wymian przedmiotów, występujących w rozwiązaniu znalezionym

algorytmu `EXACT`, który nie może działać na liczbach rzeczywistych. Algorytm `BISSA` może działać na liczbach rzeczywistych.

Tabela 1: Rezultaty uzyskane dla zestawu $(unc, 10, 1000)$.

1	2	3	4	5	6	7	8	9	10
500_1	4991	4990	1	0,020	4991	4991,828	1,828	0,037	6
500_2	**4995**	**4995**	**0**	**0,000**	4995	4995,768	0,768	0,015	6
1000_1	9968	9967	1	0,010	9970	9970,777	3,777	0,038	6
1000_2	9992	9991	1	0,010	9993	9993,523	2,523	0,025	6
5000_1	49928	49926	2	0,004	49928	49928,282	2,282	0,005	8
5000_2	49868	49864	4	0,008	49876	49876,568	12,568	0,025	6
10000_1	99808	99807	1	0,001	99813	99813,203	6,203	0,006	6
10000_2	99803	99790	13	0,013	99821	99821,231	31,231	0,031	6
50000_1	155107	155059	48	0,031	155114	155114,411	55,411	0,036	6
50000_2	155104	155091	13	0,008	155112	155112,511	21,511	0,014	8
100000_1	310292	310254	38	0,012	310294	310294,560	40,560	0,013	6
100000_2	**310282**	**310282**	**0**	**0,000**	310294	310294,664	12,664	0,004	6

Tabela 2: Indeksy przedmiotów w poszczególnych kategoriach, wyznaczone przez algorytm `BISSA` dla zadań $(unc, 10, 1000)$.

Id problemu	kat.1	kat.2	kat.3	kat.4	kat.5	kat.6	kat.7	kat.8	kat.9	kat.10
500_1	724	205	104	608	693	823	107	156	485	605
500_2	518	942	508	575	73	564	243	23	359	720
1000_1	80	109	434	91	974	778	526	387	68	29
1000_2	509	945	866	954	949	326	977	225	72	762
5000_1	226	954	230	524	581	121	563	380	40	76
5000_2	361	581	58	450	646	172	514	843	137	14
10000_1	228	757	976	986	675	113	64	76	432	482
10000_2	230	580	641	470	618	404	179	946	81	552
50000_1	499	563	631	834	771	717	840	564	273	303
50000_2	454	774	885	582	279	862	540	924	261	749
100000_1	46	97	84	644	995	54	107	700	144	419
100000_2	496	301	347	84	594	773	55	972	689	879

Tabela 3: Rezultaty uzyskane dla zestawu $(wco, 10, 10)$.

1	2	3	4	5	6	7	8	9	10
1000_1	4784	4341	443	10,205	4788	4788,78421	447,784	10,315	7
1000_2	4515	4190	325	7,757	4519	4519,13103	329,131	7,855	7
5000_1	23119	20765	2354	11,336	23127	23127,76391	2362,764	11,379	7
5000_2	23341	21922	1419	6,473	23346	23346	1424,000	6,496	6
10000_1	55220	52812	2408	4,560	55229	55229,20321	2417,203	4,577	5
10000_2	60645	59259	1386	2,339	60653	60653,89052	1394,891	2,354	7
50000_1	66373	63180	3193	5,054	66386	66386,10403	3206,104	5,075	6
50000_2	57531	54874	2657	4,842	57538	57538,74958	2664,750	4,856	6
100000_1	142720	134578	8142	6,050	142730	142730,0675	8152,067	6,058	6
100000_2	129431	124335	5096	4,099	129441	129441,2709	5106,271	4,107	6

Tabela 4: Indeksy przedmiotów w poszczególnych kategoriach wyznaczone przez algorytm `BISSA` dla zadań $(wco, 10, 10)$.

Id problemu	kat.1	kat.2	kat.3	kat.4	kat.5	kat.6	kat.7	kat.8	kat.9	kat.10
1000_1	9	8	9	2	4	4	9	1	6	9
1000_2	5	3	4	6	7	5	2	6	7	4
5000_1	10	3	9	8	7	7	6	10	6	6
5000_2	2	10	1	7	7	6	7	6	3	1
10000_1	8	5	7	9	7	5	7	7	5	9
10000_2	2	10	6	1	4	10	5	2	10	1
50000_1	8	2	2	1	3	2	2	2	4	4
50000_2	6	5	3	6	6	1	2	9	6	6
100000_1	4	5	4	10	4	7	6	1	7	7
100000_2	7	2	6	2	4	10	1	8	2	10

przez algorytm `BISSA`, na "wiceliderów", sprawdzając jednocześnie, czy następuje poprawa wartości profitu znalezionego przez algorytm `BISSA` i wybierając największy z poprawionych profitów. Dla zadania, które ma k kategorii, dla kombinacji 3-elementowej jest do wykonania $O(k^3)$ wymian i sprawdzeń. Przykładowo, dla problemu o liczbie kategorii $k = 100$ i liczbie grup 3 jest 10^6 wymian do sprawdzenia. Do testów przyjęto: dla $k = 10$ $lgrup = 5$, dla $k = 20$ $lgrup = 5$, dla $k = 100$ $lgrup = 4$ lub $lgrup = 5$, dla $k = 1000$ $lgrup = 2$. W zależności od dostępnej mocy obliczeniowej można zwiększyć wartość parametru $lgrup$.

Przykładowe wyniki testów algorytmu `SKAN-LIDER` $lgrup = 5$ dla zestawu problemów $(unc, 10, 1000)$, rozwiązanych wcześniej algorytmem `BISSA` przedstawiono w tabeli 5.

Nagłówki kolumn w tabeli (5) oznaczają:

1 – id problemu,

2 – profit dokładnego rozwiązania znaleziony przez algorytm `EXACT`,

3 – profit przybliżonego rozwiązania (LB) znaleziony przez algorytm `BISSA`,

4 – najlepszy profit uzyskany dla parametru $lgrup = 1$ (podmiana jednego wicelidera),

5 – najlepszy profit uzyskany dla parametru $lgrup = 2$ (podmiana dwóch wiceliderów),

6 – najlepszy profit uzyskany dla parametru $lgrup = 3$ (podmiana trzech wiceliderów),

7 – najlepszy profit uzyskany dla parametru $lgrup = 4$ (podmiana czterech wiceliderów),

8 – najlepszy profit uzyskany dla parametru $lgrup = 5$ (podmiana pięciu wiceliderów),

9 – numery grup z wymienionymi przedmiotami dla najlepszego wyniku,

10 – uwagi – możliwe opcje:

(a) BRAK POPR. – algorytm `SKAN-LIDER` nie poprawił wartości profitu rozwiązania przybliżonego wyznaczonego przez algorytm `BISSA`,

Tabela 5: Wyniki obliczeń algorytmem SKAN-LIDER $lgrup = 5$ dla zestawu $(unc, 10, 1000)$.

1	2	3	4	5	6	7	8	9	10
500_1	4991	4990	4990	4990	4990	4990	4990	-	BRAK POPR.
500_2	4995	4995	4995	4995	4995	4995	4995	-	BISSA – dokł.
1000_1	9968	9967	9968	9967	9967	9967	9967	(2)	DOKŁADNE
1000_2	9992	9991	9991	9992	9992	9991	9991	(3,5),(3,4,5)	DOKŁADNE
5000_1	49928	49926	49928	49926	49926	49926	49926	(4)	DOKŁADNE
5000_2	49868	49864	49864	49864	49864	49864	49864	-	BRAK POPR.
10000_1	99808	99807	99807	99807	99807	99807	99807	-	BRAK POPR.
10000_2	99803	99790	99803	99795	99790	99790	99790	(4), (4,8)	DOKŁADNE
50000_1	155107	155059	155107	155089	155078	155066	155066	(4)	DOKŁADNE
50000_2	155104	155091	155099	155091	155091	155091	155091	(5)	POPRAWA
100000_1	310292	310254	310276	310283	310275	310254	310254	(4,6)	POPRAWA
100000_2	310282	310282	310282	310282	310282	310282	310282	-	BISSA – dokł.

(b) POPRAWA – algorytm SKAN-LIDER poprawił wartość profitu rozwiązania przybliżonego wyznaczonego przez algorytm BISSA,

(c) DOKŁADNE – algorytm SKAN-LIDER wyznaczył dokładną wartość profitu optymalnego,

(d) BISSA - dokł. – algorytm SKAN-LIDER nie mógł poprawić wartość profitu, ponieważ algorytm BISSA wyznaczył dokładną wartość profitu optymalnego.

5.2 Porównanie z wynikami obliczonymi przez pakiet GUROBI

Zadania rozwiązane wcześniej zgodnie z algorytmem BISSA zostały rozwiązane przez pakiet GUROBI. Przykładowe wyniki obliczeń wykonanych przez pakiet GUROBI dla zestawu problemów $(unc, 10, 1000)$, rozwiązanych wcześniej algorytmem BISSA przedstawiono w tabeli 6.

Nagłówki kolumn w tabeli (6) oznaczają:

1 – id problemu,

2 – profit przybliżonego rozwiązania (LB) znaleziony przez algorytm BISSA,

3 – wartość profitu (UB) znaleziona przez algorytm BISSA,

4 – profit rozwiązania optymalnego wyznaczony przez pakiet GUROBI,

5 – różnica między wartościami w kolumnach 4 i 2,

6 – względna różnica (%) pomiędzy wartościami w kolumnach 4 i 2,

7 – oszacowanie od góry na wartość profitu rozwiązania optymalnego, wyznaczone przez pakiet GUROBI w węźle początkowym drzewa poszukiwań MIP (drzewo "branch and bound" – UB(GUROBI),

Tabela 6: Wyniki obliczeń wykonanych przez pakiet GUROBI dla zestawu $(unc, 10, 1000)$.

1	2	3	4	5	6	7	8	9
500_1	4990	4991,828	4991	1	0,020	4991,828	0,00013	0,000003
500_2	4995	4995,768	4995	0	0,000	4995,768	-0,00032	0,000006
1000_1	9967	9970,777	9968	1	0,010	9970,777	0,00000	0
1000_2	9991	9993,523	9992	1	0,010	9993,523	-0,00042	0,000004
5000_1	49926	49928,282	49928	2	0,004	49928,280	0,00239	0,000005
5000_2	49864	49876,568	49868	4	0,008	49876,570	-0,00202	0,000004
10000_1	99807	99813,203	99808	1	0,001	99813,200	0,00312	0,000003
10000_2	99790	99821,231	99803	13	0,013	99821,230	0,00108	0,000001
50000_1	155059	155114,411	155107	48	0,031	155114,400	0,01126	0,000007
50000_2	155091	155112,511	155104	13	0,008	155112,500	0,01089	0,000007
100000_1	310254	310294,560	310292	38	0,012	310294,600	-0,04043	0,000013
100000_2	310282	310294,664	310282	0	0,000	310294,700	-0,03573	0,000012

8 – różnica między wartościami w kolumnach 3 i 7,

9 – względna różnica (%) pomiędzy wartościami w kolumnach 3 i 7.

5.3 Testy numeryczne w wybranym zastosowaniu (dane rzeczywiste)

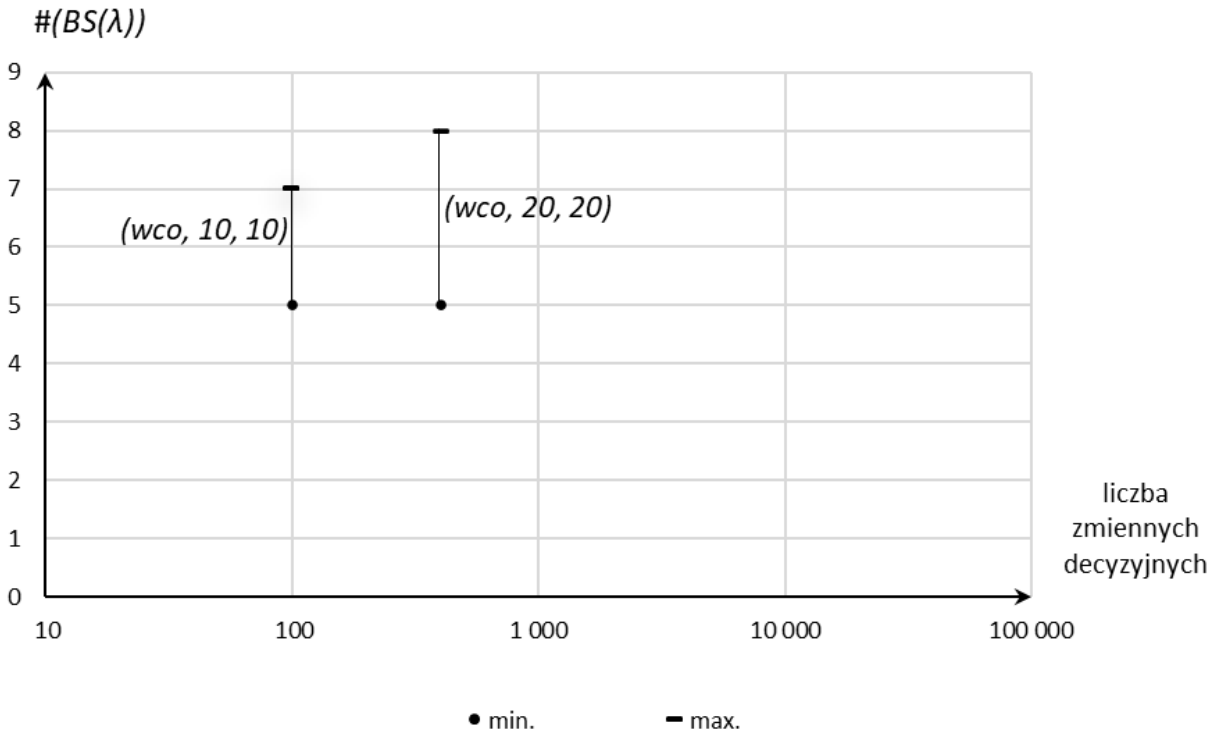
Do testu wykorzystane zostały dane rzeczywiste przedstawione w artykule [35]. Opisany w tej pracy praktyczny problem z obszaru planowanych inwestycji drogowych w rejonie metropolii Seattle w USA, został sformułowany jako zadanie $(MCKP)$, w następujący sposób: dostępnych jest 28 projektów inwestycji drogowych, każdy z tych projektów ma przypisany profit (korzystność) i koszt, każdy projekt jest przypisany do jednej z 17 kategorii. Należy wybrać najbardziej korzystną pulę 17 projektów inwestycji, po jednym z każdej kategorii, przy dostępnym budżecie, odpowiednio, w wariancie B1: 15 mld \$, w wariancie B2: 10 mld \$. W artykule [35] do obliczeń wykorzystano pakiet komercyjny LINGO.

Do rozwiązania tych praktycznych instancji problemu $(MCKP)$, zarówno w wariancie B1, jak i B2, algorytm `BISSA` potrzebował rozwiązać 3 problemy $(BS(\lambda))$.

Wektory rozwiązań $\hat{x}$ dostarczone przez algorytm `BISSA` są zgodne, odpowiednio dla budżetów B1 i B2, z wektorami rozwiązań dostarczonymi przez pakiet LINGO, przedstawionymi w artykule [35].

6 Podsumowanie, wnioski

W rozprawie zaprezentowano nową metodę rozwiązywania zagadnienia załadunku z wyborem, polegającą na zamianie ograniczenia budżetu na drugą funkcję celu. Takie przejście od zadania z jednym kryterium do zadania dwu-kryteriowego, pozwala na efektywny przegląd przestrzeni decyzyjnej, poprzez szybkie rozwiązywanie liniowego binarnego zagadnie-



Rys. 4: Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych dla liczby kategorii $k = 10$ i $k = 20$.

nia optymalizacyjnego. Jest to możliwe dzięki dekompozycji tego zagadnienia na niezależne i łatwo rozwiązywane podproblemy. Liczba problemów ($BS(\lambda)$) (16), rozwiązywanych przez algorytm `BISSA` jest względnie mała, wziąwszy pod uwagę liczbę zmiennych decyzyjnych. Nawet dla testowego zestawu $(unc, 1000, 1000)$, w którym liczba zmiennych decyzyjnych wynosi 1000000, liczba problemów ($BS(\lambda)$) (16) rozwiązywanych przez algorytm `BISSA` nie przekroczyła 14.

Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych (skala logarytmiczna) dla zadań słabo skorelowanych oraz liczby kategorii $k = 10$ i $k = 20$, została przedstawiona na rys. 4.

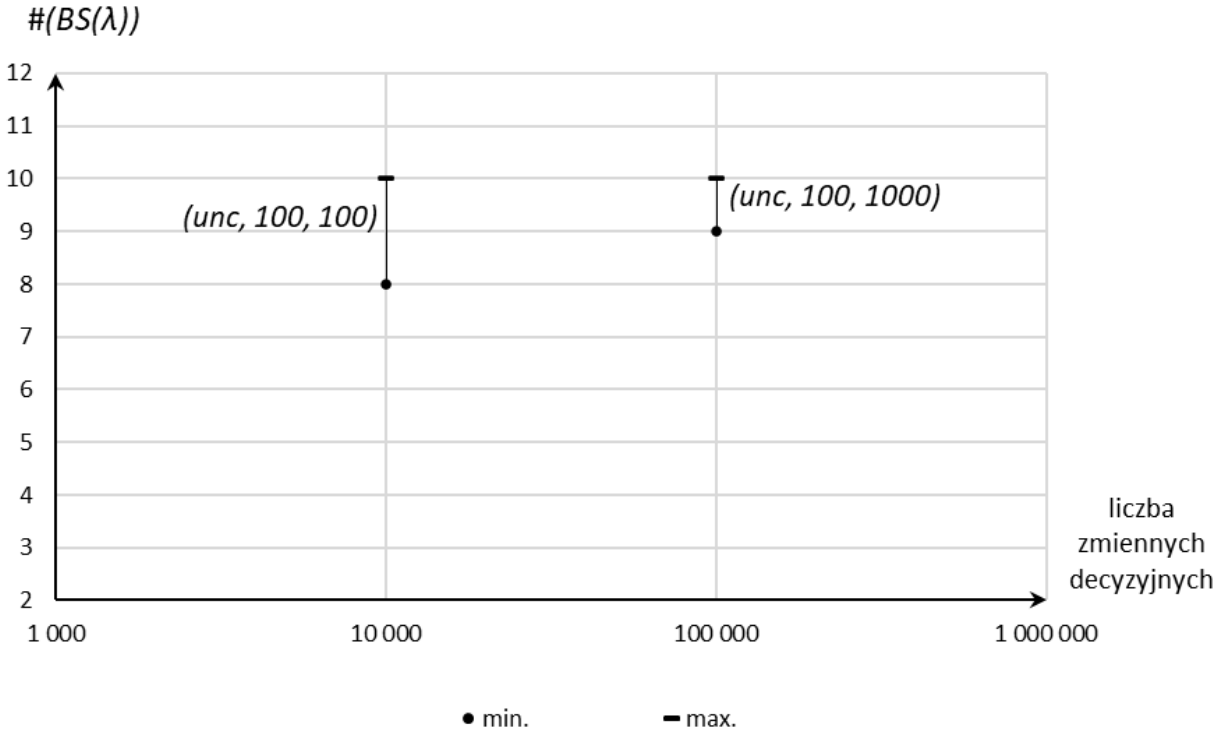
Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych (skala logarytmiczna) dla zadań nieskorelowanych oraz liczby kategorii $k = 100$, została przedstawiona na rys. 5.

Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych (skala logarytmiczna) dla zadań nieskorelowanych oraz liczby kategorii $k = 1000$, została przedstawiona na rys. 6.

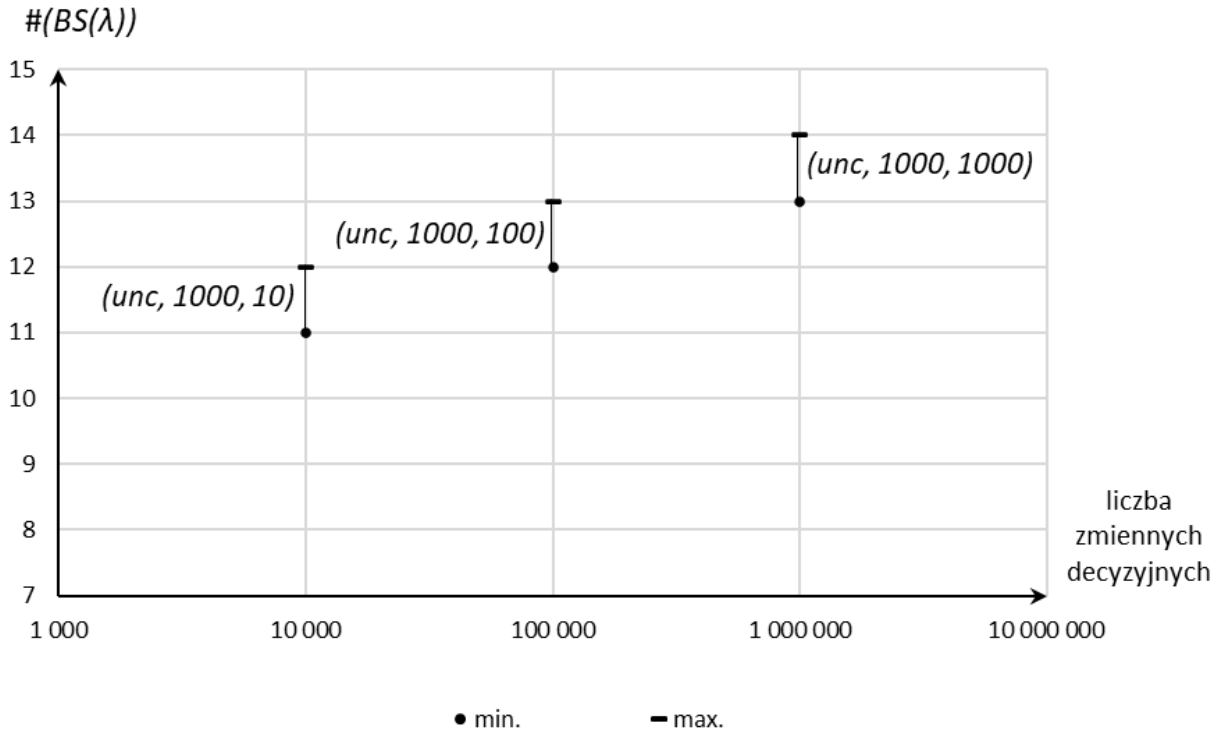
Wyniki dostarczane przez algorytm `BISSA` są porównywalne z wynikami uzyskanymi za pomocą algorytmów `EXACT`, `GREEDY`, a także z wynikami uzyskanymi za pomocą pakietu GUROBI.

Empirycznie potwierdzono, że oszacowania do góry (UB) wyznaczane przez algorytm `BISSA` są takie same, jak te uzyskiwane przez zastosowanie relaksacji ciągłej problemu ($MCKP$), a więc i relaksacji Lagrange'a. Metoda zaproponowana w rozprawie może być także porównywana do algorytmu zachłannego dla zagadnienia załadunku z wyborem, który, w ogólności, znajduje także przybliżone rozwiązanie i górne ograniczenie profitu.

Względne różnice między wartościami profitu dla rozwiązań dokładnych i przybliżo-



Rys. 5: Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych dla liczby kategorii $k = 100$.



Rys. 6: Liczba rozwiązanych problemów ($BS(\lambda)$) w zależności od liczby zmiennych decyzyjnych dla liczby kategorii $k = 1000$.

nych są niewielkie dla każdego z problemów testowych.

W niektórych przypadkach zadań testowych algorytm BISSA znajdował rozwiązania dokładne. W niektórych przypadkach zadań, po zastosowaniu algorytmu SKAN-LIDER, było możliwe znalezienie rozwiązania lepszego niż rozwiązanie wyznaczone przez algorytm BISSA, a nawet – rozwiązania dokładnego.

W przypadku instancji słabo skorelowanych danych, średnia względna różnica pomiędzy dokładnymi i przybliżonymi wartościami profitu jest nieco większa niż dla problemów nieskorelowanych. Przyczyną, dla której algorytm BISSA rozwiązuje słabo skorelowane instancje ze znacznie mniejszą liczbą zmiennych, niż dla nieskorelowanych instancji, jest to, że aby znaleźć element $\hat{x}$, należy przejrzeć cały zbiór rozwiązań S problemu $(BS(\lambda))$ (całkowity przegląd zbioru S względem wartości drugiej funkcji celu problemu (BP)). Dla słabo skorelowanych instancji, liczność zbioru S może być duża już nawet dla problemów należących do klasy $(wco, 30, 30)$. Przeprowadzono próbę eksperymentu dla problemu klasy $(wco, 30, 30)$. Dla najtrudniejszego problemu w tej klasie, liczność zbioru rozwiązań S problemu $(BS(\lambda))$ wyniosła 199 065 600. Dla większych, słabo skorelowanych problemów, ta liczba może być znacznie większa.

Należy podkreślić, że nawet w przypadku rezygnacji z przeglądu zbioru S, algorytm BISSA jest w stanie wyznaczyć co najmniej wartość oszacowania od góry (UB).

Porównanie profitu przybliżonego, obliczonego za pomocą algorytmu BISSA, oraz profitu optymalnego obliczonego przez pakiet GUROBI pokazuje bardzo małe różnice pomiędzy tymi profitami dla zadań z danymi nieskorelowanymi. Maksymalne różnice względne sięgają 0,037%. Dla zadań z danymi słabo skorelowanymi różnice te są zdecydowanie większe, zwłaszcza w przypadku mniejszej liczby zmiennych decyzyjnych. Maksymalne różnice względne sięgają w tej klasie zadań 11,336%.

Porównanie wartości oszacowania od góry (UB), obliczonych za pomocą algorytmu BISSA, oraz wartości oszacowania od góry, wyznaczonych przez pakiet GUROBI w węźle początkowym drzewa "branch and bound", pokazuje wysoką zgodność wyników obliczeń, zarówno dla zadań z danymi nieskorelowanymi, jak i dla zadań z danymi słabo skorelowanymi. Maksymalne różnice względne sięgają 0,00004%.

Wynika z tego, że zaprezentowana metoda całkiem dobrze sobie radzi w przypadku zadań z nieskorelowanymi danymi (nawet z zadaniami wielkiej skali) i w przypadku zadań ze słabo skorelowanymi danymi, i może być stosowana wszędzie tam, gdzie nie mogą być stosowane pakiety komercyjne. Algorytm BISSA znalazł już nawet możliwe praktyczne zastosowanie, opisane w literaturze [3], [4].

Inną, ważną z praktycznego punktu widzenia, zaletą algorytmów BISSA i SKAN-LIDER jest to, że są one relatywnie proste w implementacji.

W przypadku dwóch zadań na danych rzeczywistych, algorytm BISSA znalazł rozwiązania dokładne, tak samo jak pakiet LINGO.

Algorytm heurystyczny SKAN-LIDER w wielu przypadkach poprawiał rozwiązanie, znalezione przez algorytm BISSA. W niektórych przypadkach znajdował optymalne rozwiązanie.

W warunkach praktycznych zastosowań algorytmu BISSA, jakość rozwiązań wyznaczanych przez ten algorytm może być oceniana w kategoriach różnic lub względnych różnic wartości oszacowania od dołu (LB) i oszacowania od góry (UB). Przedział [(LB), (UB)], jak to pokazały przeprowadzone eksperymenty, jest relatywnie wąski.

Warto podkreślić, że algorytm BISSA podaje numery elementów w kategoriach, dających rozwiązanie przybliżone. W odróżnieniu od algorytmów opartych na programowaniu

dynamicznym, w algorytmie BISSA nie wykonuje się wstępnej selekcji, nie rejestruje się stanów dochodzenia do optymalnego rozwiązania, nie wykonuje się tzw. backtrackig-u, wskazującego rozwiązanie. Ponadto wartości profitu i kosztu poszczególnych elementów, jak również prawa strona ograniczenia budżetu, mogą być liczbami rzeczywistymi.

Przeprowadzone badania pozwoliły na zrealizowanie celu badawczego, tj. na wykorzystanie technik optymalizacji wielokryterialnej do rozwiązywania problemu ($MCKP$) oraz konstrukcję dokładnego (nie heurystycznego), efektywnego algorytmu BISSA, który znajduje dokładne rozwiązanie problemu ($MCKP$) albo: rozwiązanie przybliżone problemu ($MCKP$), wartość profitu rozwiązania przybliżonego, oszacowanie od góry na wartość optymalnego profitu problemu ($MCKP$). Ponadto skonstruowano algorytmy: dokładny SKAN-DOK, który startując z uzyskanego rozwiązania przybliżonego znajduje rozwiązanie dokładne problemu ($MCKP$) i algorytm heurystyczny SKAN-LIDER, który może poprawić uzyskane rozwiązanie przybliżone problemu ($MCKP$).

Wyniki badań teoretycznych i empirycznych potwierdzają postawioną w pracy tezę, że zastosowanie technik optymalizacji wielokryterialnej do rozwiązywania zagadnienia ($MCKP$), pozwala skonstruować relatywnie prosty algorytm, który daje wyniki porównywalne z wynikami otrzymanymi przez komercyjne pakiety do rozwiązywania zadań programowania całkowitoliczbowego (np. GUROBI).

Dalsze prace będą się skupiać na badaniu, jak algorytm BISSA zachowuje się dla mocno skorelowanych instancji danych testowych, oraz na poszukiwaniu innych niż SKAN-LIDER algorytmów dla potrzeb "skanowania" trójkąta niepewności pod kątem poprawy rozwiązania, otrzymanego algorytmem BISSA.

Widoczne są silne związki algorytmu BISSA z relaksacją Lagrange'a problemu ($MCKP$). Przeprowadzone eksperymenty pokazały, że dla zadań testowych, wartości oszacowania od góry są takie same, jak te, otrzymane za pomocą relaksacji ciągłej, a więc i za pomocą rozwiązania dualnego zadania Lagrange'a. Dalsze prace teoretyczne będą prowadzone pod kątem sprawdzenia hipotezy, że algorytm BISSA rozwiązuje *de facto* dualne zadanie Lagrange'a dla problemu ($MCKP$).

Literatura

[1] Akbar, M.M., Rahman, M.S., Kaykobad, M., Manning, E.G., Shoja, G.C.: Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. Computers & Operations Research **33**(5), 1259 – 1273 (2006). DOI http://dx.doi.org/10.1016/j.cor.2004.09.016. URL http://www.sciencedirect.com/science/article/pii/S0305054804002370

[2] Bednarczuk, E.M., Miroforidis, J., Pyzel, P.: A multi-criteria approach to approximate solution of multiple-choice knapsack problem. Computational Optimization and Applications (2018). DOI 10.1007/s10589-018-9988-z. URL https://doi.org/10.1007/s10589-018-9988-z

[3] Chanyour, T., El Ghmary, M., Hmimz, Y., Cherkaoui Malki, M.O.: Energy-efficient and delay-aware multitask offloading for mobile edge computing networks. Transactions on Emerging Telecommunications Technologies **n/a**(n/a), e3673 (2019). DOI

10.1002/ett.3673. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3673. E3673 ett.3673

[4] Chanyour, T., Hmimz, Y., EL, M., Oucamah, M.: Delay-aware and user-adaptive offloading of computation-intensive applications with per-task delay in mobile edge computing networks. International Journal of Advanced Computer Science and Applications **11** (2020). DOI 10.14569/IJACSA.2020.0110190. URL https://www.researchgate.net/publication/339076387_Delay-Aware_and_User-Adaptive_Offloading_of_Computation-Intensive_Applications_with_Per-Task_Delay_in_Mobile_Edge_Computing_Networks

[5] Chen, Y., Hao, J.K.: A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem. European Journal of Operational Research **239**(2), 313 – 322 (2014). DOI http://dx.doi.org/10.1016/j.ejor.2014.05.025. URL http://www.sciencedirect.com/science/article/pii/S0377221714004482

[6] Dudzinski, K., Walukiewicz, S.: A fast algorithm for the linear multiple-choice knapsack problem. Operations Research Letters **3**(4), 205 – 209 (1984). DOI http://dx.doi.org/10.1016/0167-6377(84)90027-0. URL http://www.sciencedirect.com/science/article/pii/0167637784900270

[7] Dudzinski, K., Walukiewicz, S.: Exact methods for the knapsack problem and its generalizations. European Journal of Operational Research **28**(1), 3 – 21 (1987). DOI http://dx.doi.org/10.1016/0377-2217(87)90165-2. URL http://www.sciencedirect.com/science/article/pii/0377221787901652

[8] Dyer, M., Kayal, N., Walker, J.: A branch and bound algorithm for solving the multiple-choice knapsack problem. Journal of Computational and Applied Mathematics **11**(2), 231 – 249 (1984). DOI http://dx.doi.org/10.1016/0377-0427(84)90023-2. URL http://www.sciencedirect.com/science/article/pii/0377042784900232

[9] Dyer, M., Riha, W., Walker, J.: A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. Journal of Computational and Applied Mathematics **58**(1), 43 – 54 (1995). DOI http://dx.doi.org/10.1016/0377-0427(93)E0264-M. URL http://www.sciencedirect.com/science/article/pii/0377042793E0264M

[10] Dyer, M.E.: An O(n) algorithm for the multiple-choice knapsack linear program. Mathematical Programming **29**(1), 57–63 (1984). DOI 10.1007/BF02591729. URL http://dx.doi.org/10.1007/BF02591729

[11] Ehrgott, M.: Multicriteria Optimization. Springer-Verlag Berlin Heidelberg (2005). URL http://www.springer.com/gp/book/9783540213987

[12] Gao, C., Lu, G., Yao, X., Li, J.: An iterative pseudo-gap enumeration approach for the multidimensional multiple-choice knapsack problem. European Journal of Operational Research pp. – (2016). DOI http://dx.doi.org/10.1016/j.ejor.2016.11.042. URL //www.sciencedirect.com/science/article/pii/S0377221716309675

[13] Glover, F.: Surrogate constraints. Operations Research **16**(4), 741–749 (1968). DOI 10.1287/opre.16.4.741. URL https://doi.org/10.1287/opre.16.4.741

[14] Gokce, E.I., Wilhelm, W.E.: Valid inequalities for the multi-dimensional multiple-choice 0-1 knapsack problem. Discrete Optimization **17**, 25 – 54 (2015). DOI https://doi.org/10.1016/j.disopt.2015.03.003. URL http://www.sciencedirect.com/science/article/pii/S1572528615000092

[15] Han, B., Leblet, J., Simon, G.: Hard multidimensional multiple choice knapsack problems, an empirical study. Computers & Operations Research **37**(1), 172 – 181 (2010). DOI http://dx.doi.org/10.1016/j.cor.2009.04.006. URL http://www.sciencedirect.com/science/article/pii/S0305054809001166

[16] Hifi, M., Michrafy, M., Sbihi, A.: Heuristic algorithms for the multiple-choice multi-dimensional knapsack problem. Journal of the Operational Research Society **55**(12), 1323–1332 (2004). DOI 10.1057/palgrave.jors.2601796. URL http://dx.doi.org/10.1057/palgrave.jors.2601796

[17] Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004). URL https://books.google.pl/books?id=u5DB7gck08YC

[18] Khan, M.S.: Quality adaptation in a multisession multimedia system: Model, algorithms, and architecture. Ph.D. thesis, University of Victoria, Victoria, B.C., Canada, Canada (1998). AAINQ36645

[19] Klamroth, K., Tind, J.: Constrained optimization using multiple objective programming. Journal of Global Optimization **37**(3), 325–355 (2007). DOI 10.1007/s10898-006-9052-x. URL http://dx.doi.org/10.1007/s10898-006-9052-x

[20] Kwong, C., Mu, L., Tang, J., Luo, X.: Optimization of software components selection for component-based software system development. Computers & Industrial Engineering **58**(4), 618 – 624 (2010). DOI http://dx.doi.org/10.1016/j.cie.2010.01.003. URL http://www.sciencedirect.com/science/article/pii/S0360835210000057

[21] Lee, C., Lehoczky, J., Rajkumar, R.R., Siewiorek, D.: On quality of service optimization with discrete QoS options. In: In Proceedings of the IEEE Real-time Technology and Applications Symposium, pp. 276–286 (1999)

[22] Martello, S., Pisinger, D., Toth, P.: New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operational Research **123**(2), 325 – 332 (2000). DOI http://dx.doi.org/10.1016/S0377-2217(99)00260-X. URL http://www.sciencedirect.com/science/article/pii/S037722179900260X

[23] Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Inc., New York, NY, USA (1990)

[24] Mathews, G.B.: On the partition of numbers. Proceedings of the London Mathematical Society **s1-28**(1), 486–490 (1896). DOI 10.1112/plms/s1-28.1.486

[25] Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers (1999). DOI 10.1007/978-1-4615-5563-6. URL http://users.jyu.fi/~miettine/book/

[26] Mohr, A.E.: Bit allocation in sub-linear time and the multiple-choice knapsack problem. In: Proceedings DCC 2002. Data Compression Conference, pp. 352–361. IEEE (2002). DOI 10.1109/DCC.2002.999973. URL http://ieeexplore.ieee.org/document/999973/. Data Compression Conference, 2002. Proceedings. DCC 2002. Conference Location: Snowbird, UT, USA, 2-4 April 2002

[27] Nauss, R.M.: The 0-1 knapsack problem with multiple choice constraints. European Journal of Operational Research **2**(2), 125 – 131 (1978). DOI http://dx.doi.org/10.1016/0377-2217(78)90108-X. URL http://www.sciencedirect.com/science/article/pii/037722177890108X

[28] Pisinger, D.: Kod programu w C. http://hjemmesider.diku.dk/~pisinger/mcknap.c (1995). (pobrany w 2016 r.)

[29] Pisinger, D.: A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operational Research **83**(2), 394 – 410 (1995). DOI https://doi.org/10.1016/0377-2217(95)00015-I. URL http://www.sciencedirect.com/science/article/pii/037722179500015I. EURO Summer Institute Combinatorial Optimization

[30] Pisinger, D.: Budgeting with bounded multiple-choice constraints. European Journal of Operational Research **129**(3), 471 – 480 (2001). DOI http://dx.doi.org/10.1016/S0377-2217(99)00451-8. URL http://www.sciencedirect.com/science/article/pii/S0377221799004518

[31] Pyzel, P.: Propozycja metody oceny efektywnosci systemow MIS. In: A. Myslinski (ed.) Techniki Informacyjne - Teoria i Zastosowania, Wybrane Problemy, vol. 2(14), pp. 59–70. Instytut Badan Systemowych PAN, Warszawa (2012)

[32] Sbihi, A.: A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. Journal of Combinatorial Optimization **13**(4), 337–351 (2007). DOI 10.1007/s10878-006-9035-3. URL http://dx.doi.org/10.1007/s10878-006-9035-3

[33] Sinha, P., Zoltners, A.A.: The multiple-choice knapsack problem. Operations Research **27**(3), 503–515 (1979). DOI 10.1287/opre.27.3.503. URL http://dx.doi.org/10.1287/opre.27.3.503

[34] Zemel, E.: An O(n) algorithm for the linear multiple choice knapsack problem and related problems. Information Processing Letters **18**(3), 123 – 128 (1984). DOI http://dx.doi.org/10.1016/0020-0190(84)90014-0. URL http://www.sciencedirect.com/science/article/pii/0020019084900140

[35] Zhong, T., Young, R.: Multiple choice knapsack problem: Example of planning choice in transportation. Evaluation and Program Planning **33**(2), 128 – 137 (2010). DOI http://dx.doi.org/10.1016/j.evalprogplan.2009.06.007. URL http://www.sciencedirect.com/science/article/pii/S014971890900041X. Challenges in Evaluation of Environmental Education Programs and Policies